Titre: Title:	Problèmes de chemins bicritères ou avec contraintes de ressource : algorithmes et applications
Auteur: Author:	Mawusé Tsévi Gabianu Vovor
Date:	1997
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Vovor, M. T. G. (1997). Problèmes de chemins bicritères ou avec contraintes de ressource : algorithmes et applications [Thèse de doctorat, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/6840/

UNIVERSITÉ DE MONTRÉAL

PROBLÈMES DE CHEMINS BICRITÈRES OU AVEC CONTRAINTES DE RESSOURCE: ALGORITHMES ET APPLICATIONS

MAWUSÉ TSÈVI GABIANU VOVOR
DÉPARTEMENT DE MATHÉMATIQUES
ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME PHILOSOPHIAE DOCTOR (Ph.D.)
(MATHÉMATIQUES DE L'INGÉNIEUR)
NOVEMBRE 1997

National Library of Canada
Acquisitions and Bibliographic Services
395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale du Canada
Acquisitions et services bibliographiques
395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

0-612-33034-6

Canada

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée:

PROBLÈMES DE CHEMINS BICRITÈRES OU
AVEC CONTRAINTES DE RESSOURCE:
ALGORITHMES ET APPLICATIONS

présentée par: VOVOR Mawusé Tsèvi Gabianu
en vue de l'obtention du diplôme de: Philosophiae Doctor
a été dûment acceptée par le jury d'examen constitué de:

M. SAVARD Gilles, Ph.D., président
Mme JAUMARD Brigitte, T.Doc., T.Hab., membre et directrice de recherche
M. HANSEN Pierre, Agr. Ens. Sup., membre et codirecteur de recherche
M. DESROSIERS Jacques, Ph.D., membre
M. RIBEIRO Celso, D. Sc., membre externe

A ma mère

REMERCIEMENTS

Une thèse de *Philosophiae Doctor* est un travail d'envergure que le candidat ne peut mener à terme seul, tout en assumant les multiples autres responsabilités de la vie de tous les jours. Je tiens à exprimer mes chaleureux remerciements à tous ceux qui, de près ou de loin, ont contribué à la réalisation de cet ouvrage. Je remercie particulièrement les Professeurs Brigitte Jaumard et Pierre Hansen pour avoir initié, encadré et financé ce travail. Mes sincères remerciements vont également au Docteur Frédéric Semet pour sa participation à l'encadrement et au financement du projet, de même qu'au Professeur Giovanni Storchi pour sa collaboration.

Je saisis également cette occasion pour exprimer ma vive reconnaissance aux Professeurs Jacques Desrosiers et François Soumis pour leurs remarques sur l'état de l'art. Les commentaires des membres du jury m'ont été très utiles, entre autres, pour l'amélioration de la présentation du texte. Qu'ils en soient ici remerciés.

Les multiples règles de la convention collective des infirmières m'ont été minutieusement expliquées par Mesdames Carolin Dagenais, Francine Brabant-Molleur, Marie-France Noël et Gianna Lepiane, de l'Hôpital Royal Victoria de Montréal. Je voudrais leur témoigner ici ma profonde gratitude. Leurs apports respectifs ont été essentiels dans le développement du modèle d'horaires d'infirmières.

Ma gratitude s'adresse aussi à tous mes collègues et amis du Groupe d'études et de recherche en analyse des décisions (GERAD), pour les multiples et fructueux échanges qui m'ont évité une perte de temps considérable. Je remercie, en particulier, mes collègues Charles Audet, Pascal Adjakplé, Stéphane Alarie, Sylvain Maréchal et Daniel Villeneuve, pour leur précieuse collaboration.

J'adresse un remerciement spécial à mon épouse pour sa patience et ses encouragements. Sa présence m'a été d'un apport extraordinaire tout au long de la thèse.

RÉSUMÉ

Nous étudions dans cette thèse divers problèmes de chemins susceptibles d'apparaître en analyse des réseaux lorsqu'un seul critère ne suffit pas pour caractériser adéquatement le chemin optimal entre deux sommets donnés. C'est notamment le cas lorsque l'on doit concilier la longueur (ou coût) du chemin avec la fiabilité ou avec les facteurs environnementaux, ou lorsque le chemin optimal doit satisfaire plusieurs contraintes de ressource.

Ces problèmes ont des applications, entre autres, en transport, en télécommunications et en confection d'horaires. Une analyse détaillée du problème d'horaires de personnel soignant est également effectuée, à titre d'application d'un problème de plus court chemin avec contraintes de ressource, plus particulièrement étudié.

Deux nouveaux critères sont introduits: celui de l'étendue minimale qui consiste à déterminer un chemin tel que la différence entre la plus grande longueur d'arc et la plus petite soit minimum, et celui du ratio minimal où l'on cherche à minimiser le rapport entre ces deux longueurs d'arc. Des algorithmes polynomiaux sont proposés pour ces problèmes, de même que pour les extensions bicritères du critère de l'étendue avec ceux, plus classiques, de la capacité ou de la longueur de chemin. Ces problèmes apparaissent parfois comme des sous-problèmes lors de l'équilibrage des chaînes de montage. C'est en particulier le cas lorsque l'on minimise le temps d'attente aux postes de travail dans le pire cas, ou lorsque l'on considère les combinaisons éventuelles de ce critère avec le temps de cycle ou avec le nombre de postes de travail.

Les algorithmes énumèrent implicitement les chemins par ordre décroissant de l'étendue ou du ratio, de manière à éviter les chemins dont la valeur est supérieure ou égale à celle du meilleur chemin connu. La procédure d'énumération est basée sur l'observation que la solution optimale, pour le critère de l'étendue, est un chemin efficace pour le problème bicritère correspondant à la recherche d'un chemin pour lequel la plus grande longueur d'arc est minimum et la plus petite est maximum.

Un nouvel algorithme, qui exploite de l'information en provenance de la source et du puits, est également présenté, pour le problème du plus court chemin bicritère avec des coûts non négatifs. Ce problème se rencontre, par exemple, en transport de matières dangereuses lorsque l'on minimise à la fois le coût de transport et la population exposée.

De nouveaux tests de dominance sont développés pour prolonger des sous-chemins efficaces à partir de la source et du puits. Ces tests sont basés sur les extensions non dominées qui sont déjà calculées et sur une approximation extérieure de l'ensemble des extensions efficaces possibles en un sommet donné. On peut ainsi éliminer plus rapidement des étiquettes ne pouvant donner des chemins efficaces de la source au puits, même si elles sont localement non dominées. En outre, une technique est proposée pour générer efficacement les chemins proprement efficaces. Cette méthode peut également être utilisée pour initialiser l'algorithme.

Les deux procédures se généralisent aisément à l'énumération de toutes les étiquettes qui sont contenues dans une fenêtre définie sur les critères. De telles bornes peuvent servir, en pratique, à exclure les solutions efficaces comportant une trop grande détérioration de l'un des critères. Des tests effectués sur des graphes aléatoires indiquent que l'algorithme se compare favorablement à la méthode de résolution par une seule extrémité, lorsque la taille ou la densité du graphe augmente.

Nous examinons également le problème de plus court chemin avec contraintes de ressource dans un graphe acyclique. Des fenêtres de ressource sont définies sur les arcs, tandis que des bornes inférieures et supérieures sont ajoutées aux sommets pour contrôler la mise à jour des consommations de ressource.

La formulation proposée n'implique pas un dédoublement des ressources quand les mises à jour ne sont pas permises. L'extension des consommations de ressource n'est pas restreinte non plus à des fonctions non-décroissantes. Il s'agit d'une généralisation du problème de plus court chemin avec fenêtres de ressource associées aux sommets, où la mise à jour des consommations ne se fait que par rapport aux seuils inférieurs des fenêtres.

Des algorithmes pseudo-polynomiaux, basés sur la programmation dynamique et sur une approche en deux phases, sont proposés pour la nouvelle formulation. La structure de l'algorithme en deux phases permet de résoudre efficacement les problèmes de réoptimisation, lorsque certains sommets sont supprimés ou sont fixés, ou lorsque les coûts changent. En outre, les calculs de complexité indiquent que la généralisation proposée n'entraîne pas une augmentation de la complexité de pire cas par rapport au cas où l'accumulation des consommations de ressource est non-décroissante. En fait, ces calculs montrent que la complexité donnée dans la littérature pour ce cas particulier est surévaluée.

La nouvelle formulation du problème de plus court chemin avec fenêtres de ressource est utilisée pour modéliser le problème de génération d'horaires réalisables pour une infirmière donnée. Les sommets du graphe correspondent aux quarts de travail et les ressources permettent de contrôler les séquences d'affectations.

Le modèle obtenu tient compte de la complexité des règles de la convention collective relatives à l'ancienneté, à la charge de travail, aux rotations et aux congés, de manière à produire des horaires réalistes. Il s'agit d'un problème pratique et complexe de cheminement, nécessitant de fréquentes réoptimisations, qui peut être traité plus efficacement par l'algorithme en deux phases que par les autres algorithmes de plus court chemin disponibles dans la littérature.

Un modèle de génération de colonnes en variables 0-1 est donné pour le problème, plus général, de la confection d'horaires pour l'ensemble du personnel soignant d'une unité de soins. Ce modèle contient, comme problème auxiliaire, celui de la génération d'horaires réalisables pour une infirmière. Le problème maître détermine une configuration d'horaires pour satisfaire les contraintes de demande tout en minimisant le coût salarial et en maximisant les préférences personnelles et l'équilibre des équipes.

Ce modèle généralise les formulations proposées dans la littérature et peut être vu comme un schéma général pour les problèmes complexes de confection d'horaires de personnel, spécialement dans le contexte des organisations opérant en continu. Il permet d'explorer implicitement la totalité de l'ensemble des horaires potentiels, contrairement à la plupart des formulations de la littérature où l'on considère des

horaires cycliques ou prédéfinis et en nombre relativement limité. Un schéma de branchement sur les variables du problème auxiliaire a été développé pour résoudre le modèle.

Les tests numériques, effectués sur des données réelles provenant de l'hôpital Royal Victoria de Montréal, confirment la capacité du modèle à prendre en compte la plupart des multiples règles utilisées en pratique dans la construction d'horaires d'infirmières. De l'avis des infirmières-chefs, les horaires générés par le programme sont de qualité au moins égale à celle des horaires produits à la main.

ABSTRACT

We study in this thesis several path problems which may arise in network analysis when a single criterion is not suitable to fully characterize an optimal path between two given vertices. This is typically the case when a compromise must be found between cost and reliability or between cost and environmental factors. Another example is when the optimal path must satisfy several resource constraints.

Such path problems have potential applications in transportation, telecommunications and staff scheduling, among others. We also present a detailed analysis of the nurse scheduling problem, as a practical example of a resource constrained shortest path problem which is more specifically studied.

Two new path problems are introduced: the minimum range problem where a path with the smallest possible range of arc lengths is to be found, and the minimum ratio problem where a path with the minimal ratio of its largest arc length to its smallest one is sought for. Polynomial algorithms are proposed for these problems, as well as for the bicriterion extensions of the minimum range criterion with those, more classic, of the capacity or of the path length. Such problems may arise in assembly line balancing when the idle time in the worst case is to be minimized as well as the cycle time or the number of workstations

Basically, the algorithms enumerate candidate paths by decreasing range or ratio order, so as to skip paths having an objective value greater or equal to the best known value. The enumeration scheme is based on the observation that the optimal path for the range criterion is efficient (i.e., Pareto-optimal) for the bicriterion path problem where the largest arc length is minimized and the shortest one is maximized.

A new algorithm, which exploits information from both ends of the network, is also presented for the bicriterion shortest path problem with non-negative arc costs. This problem may arise, for instance, in hazardous material transportation

when designing a path-finding methodology that minimizes the total length and population at risk along the optimal path. New dominance tests are developed to extend efficient subpaths from both the source and the sink. The tests are based on non-dominated extensions already computed and on an outer approximation of the set of possible efficient extensions at a given vertex. This allows to quickly discard labels that can cannot yield efficient paths from the source to the sink, even if they are locally non-dominated.

A technique is also provided to generate the extreme efficient paths and may be used to initialize the two-ended algorithm. Both procedures can readily be modified to enumerate all the efficient labels restricted to a window defined by specified lower and upper bounds on the criteria. Such bounds may be needed to discard efficient solutions involving an important deterioration of one the criteria. Numerical tests performed on random graphs indicate that the algorithm tends to outperform the one-ended labeling algorithm, when the size or the density of the network increases.

We also examine the resource constrained shortest path problem in acyclic graphs. Resource windows are associated with the arcs, while lower and upper threshold and resetting values are given at the vertices to control the updating of the resource usage. The proposed formulation does not involve resource duplication when updating is not allowed at the vertices. Resource accumultaions are neither restricted to non-decreasing functions. This is an extension of the vertex-dependent windows formulation of the resource constrained shortest path problem, where updating is allowed only for resource usage values that are smaller than the lower end of the corresponding windows.

Pseudo-polynomial algorithms, based on dynamic programming and on a two-phase approach, are presented for the problem. The specialized two-phase algorithm is efficient for reoptimization if some vertices are removed or are fixed, or if arc costs are modified. Moreover, complexity calculations indicate that the proposed generalization does not increase the worst case complexity, with respect to the the case of non-decreasing resource accumulation functions. In fact, these calculations show that the worst case complexity given in the literature for this special case is over-estimated.

The new resource constrained shortest path formulation is used to develop a model for the single individual nurse scheduling problem. The vertices correspond to the feasible shifts while resources are used to control assignment sequences. The resulting model takes into account the complexity of the collective agreement rules related to seniority, workload, rotations and days off, so as to generate realistic individual schedules. This is a complex constrained path problems, involving frequent reoptimizations, which can be more efficiently handled by the two-phase algorithm than the other shortest path algorithms available in the literature.

A 0-1 column generation model is formulated for the more general problem of finding a configuration of schedules for the whole nursing staff of a care unit. This model involves, as an auxiliary problem, that of generating individual feasible schedules. The master problem finds a configuration of individual schedules to satisfy the demand coverage constraints while minimizing salary costs and maximizing both employee preferences and team balance.

This model generalizes further the previous formulations discussed in the literature and can be viewed as a general scheme for complex personnel scheduling problems, especially in the context of organizations which operate around the clock. It also allows an implicit full exploration of the set of potential schedules, while most of the other formulations in the literature consider cyclic or predefined schedules in a relatively limited number. A specialized branching rule, on the auxiliary problem variables, is designed to solve the problem.

Numerical tests, conducted on real data from the Royal Victoria Hospital of Montreal, confirm the capacity of the model to take into account most of the many rules used in practice during the scheduling process. According to the head nurses, the quality of the schedules generated by the program are at least equivalent to that of the schedules produced by the manual approach.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SYMBOLES

Chemin v_i-v_j	Un chemin allant du sommet v_i au sommet v_j.
c_i	Coût (scalaire) du chemin v_1-v_i.
c_{ij}	Coût (scalaire) sur l'arc (v_i, v_j).
$\underline{h}_i$ et $\overline{h}_i$	Vecteurs correspondant respectivement aux seuils inférieurs et supérieurs sur l'accumulation des ressources au sommet v_i.
$\mathcal{P}_{ij}$	Ensemble des étiquettes permanentes des chemins v_i-v_j efficaces .
$\mathcal{R}$	Ensemble des ressources associées à un graphe donné.
$\mathcal{T}_{ij}$	Ensemble des étiquettes des chemins v_i-v_j temporairement efficaces.
u_{ij}	Vecteur des consommations de ressources sur l'arc (v_i, v_j).
$\underline{w}_{ij}$ et $\overline{w}_{ij}$	Vecteurs correspondant respectivement aux bornes inférieures et supérieures des fenêtres de ressources sur l'arc (v_i, v_j).
x_0	Vecteur de ressources initial (avant mise à jour) au sommet v_1.
x_i	Vecteur de ressources (après mise à jour) au sommet v_i, pour un chemin v_1-v_i.
x_{ij}	Valeur du premier critère sur l'arc (v_i, v_j).

$\underline{x}_i$ et $\overline{x}_i$	Vecteurs correspondant respectivement aux valeurs inférieures et supérieures de mise à jour des ressources au sommet v_i.
X_{ij}	Valeur du premier critère pour un chemin v_i-v_j.
(X_{ij}^k, Y_{ij}^k).	k-ème étiquette, par valeurs croissantes de x, dans $\mathcal{P}_{ij}$ à une itération donnée.
$(X_{ij}^{(r)}, Y_{ij}^{(r)})$	Etiquette de plus petite valeur lexicographique correspondant au plus court chemin v_i-v_j pour le r-ème critère ($r = 1, 2$).
y_{ij}	Valeur du deuxième critère sur l'arc (v_i, v_j).
Y_{ij}	Valeur du deuxième critère pour un chemin v_i-v_j.
φ et θ	Paramètres utilisés pour exprimer la complexité des algorithmes.
$\Psi(\cdot, \cdot)$	Opérateur de mise à jour des ressources le long d'un chemin.
$\underline{\varphi}_i$ et $\overline{\varphi}_i$	Vecteurs correspondant respectivement à des bornes inférieures et supérieures calculées sur la consommation de ressources au sommet v_i.

INTRODUCTION

L'un des problèmes les plus répandus en analyse des réseaux est la détermination d'un chemin optimal entre deux sommets donnés. Les objectifs souvent considérés sont la longueur, la durée, le coût, la capacité, la fiabilité ou le risque. Dans bien des cas, ces problèmes se ramènent à la recherche d'un plus court chemin ou d'un chemin de capacité maximale entre les deux sommets. Certaines situations, par exemple en équilibrage des chaînes de montage, nécessitent cependant d'autres critères tels que l'étendue du chemin ou son ratio, i.e., la différence ou le rapport entre la longueur de l'arc le plus long et celle de l'arc le plus court.

Des combinaisons bicritères peuvent également être requises pour modéliser des situations réelles où plus d'un critère doit être simultanément considéré lors de la recherche d'un chemin entre deux points donnés. Par exemple, le coût et la fiabilité sont tous les deux importants dans les réseaux de télécommunications, les facteurs économiques et écologiques doivent être simultanément considérés dans la construction d'une autoroute, de même que le profit et le risque dans les projets d'investissement qui se ramènent à des problèmes de chemin.

Dans beaucoup de ces applications, il est souvent suffisant de résoudre le problème du plus court chemin avec la condition supplémentaire que la valeur de chacun des autres critères soit contenue dans un intervalle spécifique pour chaque arc utilisé. Ces critères supplémentaires peuvent, dans ce cas, être interprétés comme des ressources consommées le long du chemin. Des mises à jour peuvent être nécessaires aux sommets, lorsque la valeur d'une ressource dévie de certains seuils fixés à l'avance.

Par exemple, en confection d'horaires de personnel, l'employé ne peut passer des affectations de jour à celles de nuit qu'après avoir, entre autres, reçu consécutivement un nombre de quarts de jour respectant un minimum et un maximum prédéfinis. Une telle rotation entre affectations de jour et de nuit nécessite souvent un réajustement du compteur d'affectations consécutives de même type.

Ces problèmes de chemins bicritères ou avec contraintes de ressource apparaissent, soit directement ou comme problèmes auxiliaires, dans de multiples applications en analyse des réseaux. Dans le cadre de cette thèse, nous proposons divers algorithmes pour les critères de l'étendue et du ratio, ainsi que pour les extensions bicritères de l'étendue avec le critère du plus court chemin ou avec celui de la capacité maximale.

Un nouvel algorithme, utilisant des informations en provenance des deux extrémités du chemin, est également proposé pour le problème du plus court chemin bicritère, plus courant et plus difficile. Nous examinons, en outre, une nouvelle extension du problème du plus court chemin avec fenêtres de ressource sur un réseau acyclique. L'algorithme proposé pour ce problème est particulièrement bien adapté lorsque le problème doit être résolu plusieurs fois de suite, après des modifications n'impliquant pas les consommations de ressource, telles que l'interdiction ou l'imposition de certains sommets ou arcs.

Ce dernier point est illustré par la résolution d'un modèle de programmation linéaire généralisée pour la confection d'horaires de personnel soignant. Les variables du problème maître de ce modèle correspondent à des horaires individuels qui sont construits en résolvant un problème auxiliaire de génération de colonnes. Le problème auxiliaire est formulé comme un problème de plus court chemin avec fenêtres de ressource dans un réseau acyclique où les sommets correspondent à des affectations. Diverses ressources sont utilisées dans le modèle pour contrôler les séquences d'affectations réalisables.

Nous discutons, au chapitre 1, les formulations et la littérature relatives aux différents problèmes de chemins considérés dans la thèse. La problématique associée à la confection d'horaires de personnel soignant est également présentée dans ce chapitre à titre d'application pratique d'un problème complexe de cheminement. Le chapitre 2 décrit les algorithmes de chemins avec étendue ou ratio minimum, tandis que les chapitres 3 et 4 sont respectivement consacrés à la présentation de l'algorithme du plus court chemin bicritère et à celle de l'algorithme de plus court chemin avec fenêtres de ressource. Les détails de la modélisation et de la résolution du problème d'horaires sont abordés dans le chapitre 5, de même qu'une discussion des résultats.

CHAPITRE 1

PROBLÉMATIQUE ET BIBLIOGRAPHIE

1.1 Problèmes de chemins avec étendue ou ratio minimum

Nous considérons deux nouveaux problèmes de chemins dans les graphes. Le premier consiste à trouver un chemin d'un sommet v_1 à un sommet v_n tel que l'étendue, i.e., la différence entre la longueur de l'arc le plus long et celle de l'arc le plus court, soit la plus petite possible. Dans le deuxième, on recherche un chemin pour lequel le rapport de la longueur de l'arc le plus long à celle de l'arc le plus court soit minimum. Deux extensions bicritères de ces problèmes sont également étudiées.

1.1.1 Problèmes à un critère

Soit $G = (V, A)$ un graphe orienté avec $n = |V|$ sommets $v_1, v_2, ..., v_n$, et $m = |A|$ arcs. Considérons deux sommets distincts v_1 et v_n de G et l'ensemble $\mathcal{P}$ des chemins de v_1 à v_n (ou chemins v_1-v_n). Nous supposons que $\mathcal{P} \neq \emptyset$ et que la longueur (ou coût) c_{ij}, de tout arc $(v_i, v_j) \in A$, est un entier positif.

Les problèmes de cheminement dans les graphes ont été très largement étudiés (voir par exemple Gallo *et al.* [31], Gallo et Pallottino [30], [29], pour des revues bibliographiques). Les objectifs les plus fréquemment considérés sont la longueur

minimale (par exemple, Dijkstra [25]), la fiabilité maximale (exemple, Frisch [28]) et la capacité maximale (exemple, Punnen [54]). Les deux nouveaux critères considérés dans cette section sont présentés ci-après:

- MINRANGE: trouver un chemin v_1-v_n dans G tel que la différence entre la plus grande longueur d'arc et la plus petite soit minimum, i.e.,

$$\min_{P \in \mathcal{P}} \left(\max_{(v_i,v_j) \in P} c_{ij} - \min_{(v_i,v_j) \in P} c_{ij} \right), \tag{1.1}$$

- MINRATIO: trouver un chemin v_1-v_n dans G tel que le ratio de la plus grande longueur d'arc sur la plus petite soit minimale, i.e.,

$$\min_{P \in \mathcal{P}} \frac{\max_{(v_i,v_j) \in P} c_{ij}}{\min_{(v_i,v_j) \in P} c_{ij}}. \tag{1.2}$$

Les deux critères expriment une préférence pour un équilibre dans la distribution des longueurs d'arcs le long du chemin optimal. L'exemple suivant, en équilibrage des chaînes de montage (voir Baybars [7] pour les définitions), constitue une application potentielle de ces problèmes. Considérons le diagramme de précédence de k tâches avec des temps d'exécution $a_1, a_2, ..., a_k$ et supposons que l'ordre dans lequel ces tâches doivent être exécutées a une flexibilité limitée. Notons que les sommets, dans le diagramme de précédence, correspondent aux tâches et les arcs représentent les relations de précédence (voir la figure 1.1 pour un petit exemple).

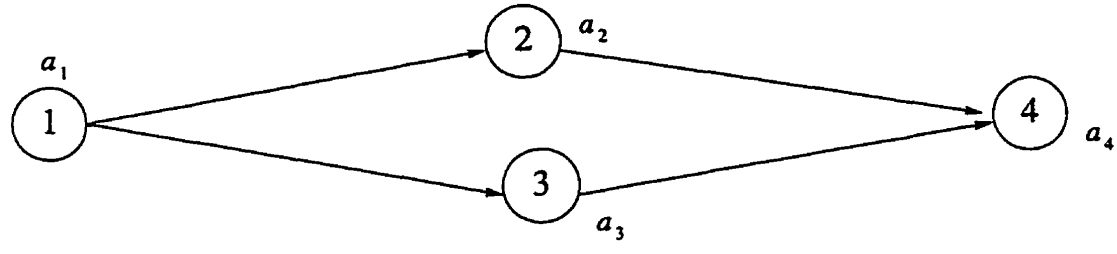


Figure 1.1 – *Exemple de diagramme de précédence*

Considérons maintenant le graphe G (figure 1.2) dans lequel les arcs correspondent aux tâches associées à un même poste de travail et les sommets aux ensembles de tâches déjà effectuées (comme dans les réseaux PERT). La longueur d'un arc est égale à la somme des durées des tâches qui le définissent. Deux sommets quelconques sont reliés par des arcs consécutifs correspondant à toutes les possibilités permises par le diagramme de précédence. Des arcs supplémentaires existent entre chaque paire de sommets qui sont des extrémités de sous-chemins pour lesquels le temps total d'exécution des tâches (la longueur du sous-chemin) n'excède pas une borne prédéfinie sur le temps de cycle. La longueur d'un tel arc est égale à la longueur du sous-chemin déjà existant entre les sommets impliqués.

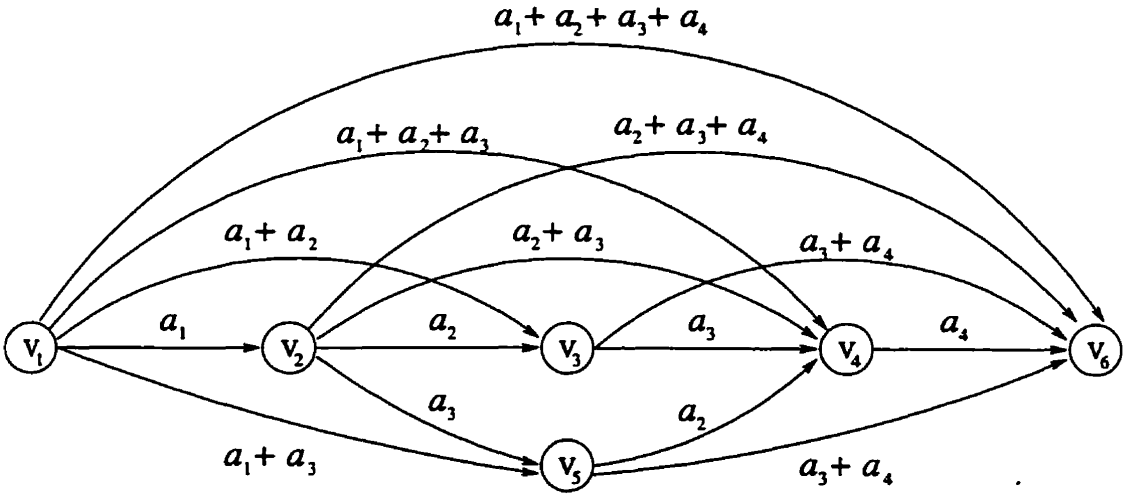


Figure 1.2 – *Exemple de graphe $G = (V, A)$ pour une chaîne de montage*

Le fonctionnement de la chaîne correspond à un chemin v_1-v_n, où aucune tâche n'est accomplie au sommet v_1 et toutes les tâches sont terminées au sommet v_n. Les tâches effectuées aux postes de travail le long de la chaîne correspondent à celles qui sont associées aux arcs du chemin. Le temps de cycle est égal à la plus grande longueur d'arc sur le chemin. La minimisation de l'étendue correspond donc à celle du temps d'attente aux postes de travail dans le pire cas. Minimiser le ratio des longueurs d'arcs revient à minimiser le pourcentage de temps d'attente aux postes de travail dans le pire cas.

Nous n'avons trouvé aucune mention du critère de l'étendue dans la littérature. Cependant, différents objectifs impliquant un ratio où au moins l'un des critères est linéaire, ont été étudiés, soit seuls ou dans des problèmes de chemins bicritères. Dans ces problèmes de ratio, deux valeurs différentes sont associées à chaque arc (par exemple la longueur et le coût ou bien le coût et la capacité), au lieu d'une seule comme dans MINRANGE et MINRATIO.

Dantzig *et al.* [16] considèrent ainsi le problème de la détermination d'un cycle qui minimise le ratio de la somme des longueurs d'arcs sur celle des durées (de la traversée) des arcs, alors que Tung [56] étudie la recherche d'un chemin élémentaire correspondant à cet objectif. Un problème bicritère impliquant cet objectif et celui de la capacité maximum est étudié dans Martins [48]. Le ratio de la somme des coûts des arcs (ou de leurs longueurs) sur la plus petite capacité des arcs est également traité par ce dernier auteur dans [46]. Une étude du problème de la détermination d'un chemin minimisant le ratio de la somme des coûts des arcs sur le produit de leurs fiabilités est effectuée dans Ahuja [2].

Nous proposons, au deuxième chapitre de cette thèse, des algorithmes pour les problèmes MINRANGE et MINRATIO. Ces algorithmes énumèrent, fondamentalement, des chemins par ordre décroissant de l'étendue ou du ratio, de manière à éviter les chemins dont la valeur est supérieure ou égale celle du meilleur chemin connu.

Un chemin efficace (dit aussi non dominé ou Paréto-optimal) pour un problème bicritère se définit comme un chemin P tel qu'il n'existe aucun autre chemin P' pour lequel l'un des critères est meilleur sans que l'autre ne soit moins bon. Ainsi, pour le problème bicritère MINMAX-MAXMIN correspondant à la recherche d'un chemin P pour lequel la plus grande longueur d'arc, $\bar{c}(P)$, est minimum et la plus petite longueur d'arc, $\underline{c}(P)$, est maximum, un chemin P_{opt} est efficace pour le problème MINMAX-MAXMIN s'il n'existe aucun autre chemin $P \in \mathcal{P}$ avec $\bar{c}(P) < \bar{c}(P_{opt})$ et $\underline{c}(P) \geq \underline{c}(P_{opt})$ ou avec $\bar{c}(P) \leq \bar{c}(P_{opt})$ et $\underline{c}(P) > \underline{c}(P_{opt})$. La procédure d'énumération, utilisée dans les algorithmes pour les problèmes MINRANGE et MINRATIO, est basée sur l'observation que la solution optimale est un chemin efficace pour le problème bicritère MINMAX-MAXMIN.

Cette procédure est similaire à celle qui consiste à résoudre un problème de chemin à un critère en déterminant un sous-ensemble de chemins efficaces pour un problème bicritère associé. Un exemple de cette approche est utilisée dans Martins [46] pour déterminer un chemin qui minimise le rapport coût/capacité, en utilisant le problème bicritère MINSUM-MAXMIN où l'on recherche un chemin de coût total minimal et dont la (plus petite) capacité est maximale (voir, par exemple, Hansen[36] ou Martins [48]). Une méthode semblable est également utilisée dans Ahuja [2] pour trouver un chemin pour lequel le rapport coût/fiabilité est minimal, en utilisant un algorithme qui résout le problème bicritère MINSUM-MINSUN consistant à déterminer un chemin pour lequel le coût et la fiabilité sont minimum.

Notons que les graphes orientés considérés dans la formulation des problèmes MINRANGE et MINRATIO contiennent, comme cas particuliers, les graphes non-orientés (dans lesquels $(v_i, v_j) \in A$ implique $(v_j, v_i) \in A$)). Pour ces graphes, des algorithmes de meilleures complexités peuvent être obtenus en exploitant un récent résultat de Punnen [54] pour le problème de chemin de capacité maximale. Nous examinons dans la prochaine section quelques extensions bicritères des problèmes MINRANGE et MINRATIO.

1.1.2 Extensions bicritères

Il est souvent pertinent d'utiliser plus d'un critère pour déterminer le meilleur chemin dans un graphe. Une liste de problèmes de chemins bicritères avec des analyses de complexité et divers algorithmes spécialisés peut être trouvée dans Hansen [36]. Une discussion sur l'utilisation de fonctions d'utilité pour résoudre les problèmes de chemins bicritères est également présentée dans Henig [38].

Dans le but de donner le maximum de flexibilité au décideur lorsque deux critères sont considérés, il est nécessaire de trouver tous les chemins efficaces. Il pourrait cependant y avoir un nombre exponentiel de chemins v_1-v_n ayant la même valeur pour les deux critères. On cherchera donc à déterminer plutôt un ensemble $\mathcal{P}^*$ de chemins v_1-v_n efficaces non équivalents, i.e., un ensemble tel qu'aucun chemin

dans $\mathcal{P}^*$ n'est dominé ni équivalent à un autre chemin de $\mathcal{P}^*$ et aucun chemin dans $\mathcal{P}^*$ ne domine ni n'est équivalent à un autre chemin de $\mathcal{P}^*$.

En guise d'application, considérons de nouveau le problème d'équilibrage de chaîne de montage discuté à la section précédente. Rappelons que le temps de cycle est la plus grande longueur d'arc sur le chemin v_1-v_n choisi. Ce temps est un paramètre important puisqu'il détermine le taux de production et contraint l'ensemble des tâches qui pourraient être exécutées le long de la chaîne.

Ainsi, il peut être intéressant d'étudier le problème bicritère MINRANGE-MINMAX pour le graphe G du problème d'équilibrage de chaîne de montage. Cela revient à considérer simultanément le critère du taux de production et du temps d'attente dans le pire cas.

On obtient un problème plus difficile lorsque le critère de l'étendue (ou du ratio) est considéré en même temps que celui de la longueur totale du chemin. Ce dernier critère a la même valeur pour tout les chemin dans l'exemple de la chaîne de montage. Il correspond au temps total de traitement le long de la chaîne.

Ainsi, si une valeur constante égale, par exemple, à une borne supérieure sur le temps de cycle, est ajoutée à toutes les longueurs d'arc, alors tout chemin v_1-v_n aura une longueur égale au temps total de traitement plus le produit de la constante et du nombre d'arcs sur le chemin (i.e., le nombre de postes de travail sur la chaîne). Le problème MINRANGE-MINSUM revient donc à considérer le critère du nombre de postes de travail et celui du temps d'attente dans le pire cas.

Un algorithme est décrit au chapitre 2 pour ce problème ainsi que pour le problème MINRANGE-MINMAX. La prochaine section est consacrée à l'exposé d'un problème bicritère plus classique, le problème MINSUM-MINSUM, où l'on considère simultanément la longueur totale du chemin pour deux critères différents.

1.2 Le problème de plus court chemin bicritère

1.2.1 Description du problème

Soit de nouveau un graphe orienté $G = (V, A)$ avec $n = |V|$ sommets $v_1, v_2, ..., v_n$ et $m = |A|$ arcs. A la différence de la section précédente, on associe, à chaque arc $(v_i, v_j) \in A$, deux valeurs non négatives (x_{ij}, y_{ij}). Les sommets v_1 et v_n correspondent à la source et au puits, respectivement.

Un chemin allant d'un sommet v_i à un sommet v_j sera appelé chemin v_i-v_j et sa valeur sera représentée par l'étiquette bicritère (X_{ij}, Y_{ij}). Chaque composante de l'étiquette est égale à la somme des valeurs correspondant au critère associé, sur tous les arcs du chemin. L'étiquette de plus petite valeur lexicographique correspondant au plus court chemin v_i-v_j pour le r-ème critère $(r = 1, 2)$ sera notée $(X_{ij}^{(r)}, Y_{ij}^{(r)})$.

Rappelons qu'une étiquette (X_{ij}, Y_{ij}), associée à un chemin v_i-v_j, est efficace (ou non dominée) s'il n'existe aucune étiquette (X'_{ij}, Y'_{ij}), également associée à un chemin v_i-v_j, telle que $X'_{ij} \leq X_{ij}$ et $Y'_{ij} \leq Y_{ij}$ avec l'inégalité stricte dans au moins un des deux cas. Un chemin correspondant à une étiquette efficace est aussi dit efficace, et vice versa.

On peut remarquer que plusieurs chemins efficaces peuvent correspondre à une même étiquette. Étant donné un sommet v_i, un chemin v_1-v_i ou un chemin v_i-v_n, dont le prolongement ne peut donner un chemin v_1-v_n efficace, sera dit non prometteur.

Le problème de plus court chemin bicritère consiste à déterminer un chemin v_1-v_n pour lequel la valeur totale de chacun des critères est minimale. Très souvent, un tel chemin n'existe pas et le problème se ramène à l'énumération des chemins v_1-v_n efficaces. Ce problème est susceptible d'apparaître dans diverses applications en analyse des réseaux, lorsqu'un seul objectif ne suffit pas pour caractériser adéquatement le problème d'optimisation sous-jacent.

1.2.2 Revue de la littérature

Le problème de transport de matières dangereuses discuté dans Chin et Cheng [12] constitue un exemple d'application du problème de plus court chemin bicritère. Les deux critères considérés par les auteurs sont respectivement la distance totale parcourue et la population exposée à l'intérieur d'une bande de largeur fixe le long du trajet.

Une variante du problème consiste à utiliser une fonction d'utilité, qui combine les deux critères, pour trouver un chemin optimal. Lorsque la fonction d'utilité est linéaire ou lexicographique, une solution existe par programmation dynamique (voir, par exemple, Loui [44]). Une méthode de recherche linéaire unimodale et un algorithme d'évaluation et de séparation progressives basé sur la génération des k-ème plus courts chemins, sont proposés dans Henig [38] pour calculer des bornes sur la valeur optimale d'une fonction d'utilité quasi-concave.

Notons cependant que, pour des fonctions d'utilité quelconques, le problème de plus court chemin bicritère n'admet pas toujours de solution par la programmation dynamique. On est alors généralement amené à énumérer les chemins efficaces avant de sélectionner la solution optimale (voir Loui [44]). En outre, une fonction d'utilité n'est pas toujours explicitement bien définie en pratique.

Nous nous intéresserons, dans cette étude, au problème de génération de l'ensemble des étiquettes efficaces correspondant à des chemins v_1-v_n. Un tel ensemble défini un ensemble minimal complet de chemins v_1-v_n efficaces, i.e., contenant exactement un chemin pour chaque étiquette efficace. Ce problème est difficile dans le pire cas: le nombre d'étiquettes efficaces croît exponentiellement en fonction de la taille du réseau (Hansen [36]).

En outre, dans l'espace des étiquettes, toutes les étiquettes efficaces du problème de plus court chemin bicritère ne sont pas des points extrêmes de l'enveloppe convexe des étiquettes correspondant aux chemins v_1-v_n du graphe (voir, par exemple, White [63]). Les étiquettes associées à ces points extrêmes seront appelées étiquettes efficaces extrêmes. Ainsi, une approche paramétrique résolvant successivement des problèmes

de plus court chemins pour un objectif obtenu par combinaison linéaire des deux critères, ne permettra pas toujours de trouver toutes les étiquettes efficaces (voir White [63], entre autres).

Un algorithme d'étiquetage est proposé dans Hansen [36] pour le problème considéré. L'auteur décrit également une méthode d'approximation permettant de déterminer les étiquettes efficaces pour un niveau de précision donné. Cette méthode est polynomiale en fonction de la taille du graphe et de l'inverse de l'erreur relative maximale permise sur la valeur de chaque critère pour un chemin v_1-v_n. Un algorithme, basé sur une méthode de k-ème plus court chemin, est aussi décrit dans Climaco et Martins [13] pour générer tous les chemins efficaces, même quand il en existe plusieurs pour une même étiquette.

Henig [38] suggère trois méthodes pour générer l'ensemble des étiquettes efficaces extrêmes. Une des méthodes est une procédure d'étiquetage permanent des chemins et les deux autres impliquent la résolution successive de problèmes paramétriques de plus court chemin, en vue de générer les étiquettes efficaces extrêmes par ordre croissant de l'un des critères.

Une approche paramétrique similaire, utilisant un argument du type simplexe pour ajuster les paramètres, suivie de la procédure d'étiquetage de Hansen [36], est présentée dans Mote *et al.* [51]. Les auteurs rapportent des résultats de calcul indiquant qu'une telle procédure combinée est significativement plus rapide que la méthode du k-ème plus court chemin et meilleure que la méthode d'étiquetage pour les problèmes avec une corrélation positive entre les deux critères. Les tests ont été effectués sur des réseaux générés aléatoirement, ayant 1000 sommets et jusqu'à 10000 arcs avec des valeurs entières de 1 à 200 pour chaque critère, ainsi que sur des réseaux quadrillés contenant 400 sommets et jusqu'à 1520 arcs pour des longueurs d'arcs entières de 1 à 100.

Cependant, comme indiqué dans Henig [38], les méthodes paramétriques du type "simplexe" sont hautement sensibles à la dégénérescence de la base. Ceci se produit lorsque l'arc entrant dans la base courante n'entraîne pas de modification de l'étiquette résultante. Plusieurs itérations sont alors nécessaires pour qu'une nouvelle

étiquette efficace extrême puisse être trouvée. Henig [38] propose une procédure alternative qui nécessite le calcul de plus courts chemins successifs un nombre de fois égal à deux fois le nombre de chemins efficaces extrêmes.

1.2.3 Une approche par les deux extrémités

La méthode d'étiquetage a l'inconvénient d'entraîner une croissance rapide du nombre d'étiquettes efficaces correspondant à des sous-chemins qui ne donneront pas de chemins v_1-v_n efficaces. Un test de dominance est présenté dans Tung et Chew [58] pour réduire le nombre de ces étiquettes non prometteuses. Le test utilise un chemin fictif dont l'étiquette est constituée par les valeurs des plus courts chemins d'un sommet donné au sommet v_n, pour chacun des critères. L'algorithme résultant peut être considéré comme une méthode en deux phases pour résoudre le problème par les deux extrémités du réseau.

Peu de travaux existent dans la littérature sur la résolution du problème de plus court chemin bicritère par les deux extrémités. Une analyse critique de quelques algorithmes utilisant une approche par les deux extrémités pour le problème de plus court chemin à un critère peut être trouvée dans Dreyfus [27]. Un algorithme plus récent dans cette catégorie est discuté dans Jeyaratnam [42].

Les techniques utilisées consistent à prolonger les sous-chemins à la fois à partir des sommets v_1 et v_n, de manière à obtenir une procédure en une phase. Cependant, comme rapporté dans Dreyfus [27], l'une des difficultés majeures d'une telle approche symétrique est le développement de tests d'arrêt et de dominance corrects et efficaces. En outre, la complexité de pire cas de ces algorithmes est en général moins bonne que celle de l'algorithme de Dijkstra [25] qui résout le problème par un seul extrémité (voir Dreyfus [27]).

Cependant, dans le cas bicritère, une approche symétrique par les deux extrémités pourrait significativement réduire le nombre total d'étiquettes examinées, étant donné qu'environ la moitié seulement des sommets serait considérée autant dans la phase

"en avant" que celle "en arrière". Cette observation constitue l'une des motivations pour le développement de l'algorithme proposé au chapitre 3 de cette thèse pour résoudre le problème de plus court chemin bicritère par les deux extrémités.

L'algorithme est initialisé en déterminant, pour chaque sommet et pour chaque critère, les étiquettes de plus petites valeurs lexicographiques correspondant aux plus courts chemins de v_1 au sommet considéré et de celui-ci à v_n. Une extension de la procédure d'initialisation est aussi discutée. Elle implique le calcul des chemins efficaces extrêmes, non nécessairement suivant l'ordre croissant de l'un des critères.

Cette procédure n'est pas susceptible de dégénérescence et est particulièrement bien adaptée pour trouver les étiquettes efficaces extrêmes restreintes à un rectangle défini par une borne inférieure et une borne supérieure sur chacun des critères. De telles bornes peuvent être spécifiées par le décideur dans le but de limiter la recherche de solutions efficaces à une région donnée de l'espace des solutions. Ceci permet d'exclure, éventuellement, les solutions correspondant à une trop grande détérioration de l'un des critères.

De nouveaux tests de dominance, plus forts que ceux de Tung et Chew [58], et utilisant des approximations extérieures de l'ensemble des étiquettes efficaces à un sommet donné, sont incorporés dans l'algorithme. La performance numérique de cette approche est comparée à celle de l'algorithme d'étiquetage de Hansen [36]. Les résultats sont discutés au chapitre 3.

Dans la prochaine section, nous décrivons un autre problème de plus court chemin impliquant plus d'un critère. Contrairement au problème bicritère, il s'agira de minimiser un seul des critères, les autres étant contrôlés par des fenêtres définies sur les arcs et aux sommets. Cependant, les algorithmes seront aussi basés sur le principe d'utilisation de l'information en provenance de la source et du puits.

1.3 Un problème de plus court chemin avec fenêtres de ressource

On considère, dans cette partie, le problème de trouver un chemin de coût (ou de longueur) minimum dans un réseau acyclique où la traversée de chaque arc implique la consommation d'une ou de plusieurs ressources. Un arc donné ne peut être utilisé que si les ressources cumulées jusqu'à l'origine de l'arc respectent des fenêtres définies par des bornes inférieures et supérieures sur cet arc.

On permet une correction ou remise à jour de la consommation des ressources aux sommets. Ainsi, si la ressource cumulée excède un seuil supérieur prédéfini, elle est ramenée à une valeur supérieure de mise à jour également prédéfinie. De manière similaire, un seuil et une valeur de mise à jour inférieures sont définis pour chaque ressource et pour chaque sommet.

Une description formelle du problème est présentée à la section 1.3.1. Ce problème se rencontre, par exemple, en confection d'horaires de personnel où les sommets représentent les affectations et un chemin correspond à une séquence d'affectations sur un horizon de planification donné. Les ressources peuvent correspondre alors aux nombres d'affectations consécutifs de jour, de soir ou de nuit. Un exemple de cette application est fourni à la section 1.4.

Divers problèmes de plus court chemin avec quelques contraintes additionnelles ont été considérés dans la littérature par plusieurs auteurs incluant Joksch [43], Saigal [59], Minoux [50], Handler et Zang [35], Hansen [36], Aneja, Aggarwal et Nair [3], Jaffe [41], Martins [47], de même que Ribeiro et Minoux [55]. Une revue bibliographique de ces travaux peut être trouvée dans Beasley et Christofides [8].

Les méthodes de résolution proposées par ces auteurs sont basées sur des techniques telles que la programmation dynamique (exemple, [43], [59]), l'étiquetage (exemple, [36], [47]), la relaxation lagrangienne combinée avec des algorithmes de k-èmes plus courts chemins (exemple, [35]) or ou avec une procédure d'évaluation et de séparation progressives (exemple, [8]). Une heuristique, basée sur la relaxation

lagrangienne et un calcul de plus courts chemins paramétriques, est également discutée dans Ribeiro et Minoux [55], pour le cas où le chemin cherché est élémentaire et la consommation totale de la ressource est doublement contrainte.

Ces algorithmes sont conçus pour des contraintes additionnelles actives seulement au sommet terminal v_n. Cependant, dans le problème de plus court chemin que nous considérons, de telles contraintes peuvent être présentes à n'importe quel sommet et ne sont pas nécessairement linéaires.

Un problème, voisin de celui considéré ici, est discuté dans Desrochers [19] (voir aussi Desrosiers, Dumas, Solomon et Soumis [22]), où la mise à jour n'est permise que pour des valeurs de ressource plus petites que des seuils inférieurs prédéfinis. Les fenêtres de ressource sont associées aux sommets et coïncident avec celles des seuils et valeurs de mise à jour. Il s'agit d'une généralisation multidimensionnelle du problème de plus court chemin avec fenêtres de temps, discuté dans Desrosiers, Pelletier et Soumis [23] et dans Desrosiers, Soumis et Desrochers [24], comme problème auxiliaire du problème des multiples voyageurs de commerce avec fenêtres de temps.

Un algorithme de programmation dynamique est également décrit dans Ioachim *et al.* [40] pour le cas où des coûts linéaires, éventuellement décroissants en fonction de la ressource, sont appliqués aux sommets du plus plus court chemin avec fenêtres de temps. Une formulation générale de ces problèmes de chemins avec contraintes de ressource est présentée dans Desaulniers *et al.* [18] mais aucun algorithme n'est décrit pour ce cas général.

La résolution du problème des multiples voyageurs de commerce avec fenêtres de temps, nécessite de résoudre plusieurs fois le problème auxiliaire, dans un processus de génération de colonnes, après avoir modifié les coûts sur les arcs. Ainsi, un algorithme efficace est requis pour la résolution répétitive du problème de plus court chemin avec contraintes. Desrochers et Soumis décrivent, pour ce problème, un algorithme d'étiquetage permanent et une procédure de réoptimisation primale-duale dans [20] et [21] respectivement. Nous présentons, ci-après, une description formelle de la nouvelle extension du problème de plus court chemin avec fenêtres de ressource.

1.3.1 Formulation du problème

On considère un réseau orienté acyclique avec $n = |V|$ sommets, $v_1, v_2, ..., v_n$, en ordre topologique et $m = |A|$ arcs, où les sommets v_1 et v_n sont respectivement la source et le puits. Un ensemble $\mathcal{R}$ de ressources à valeurs discrètes est associé à G.

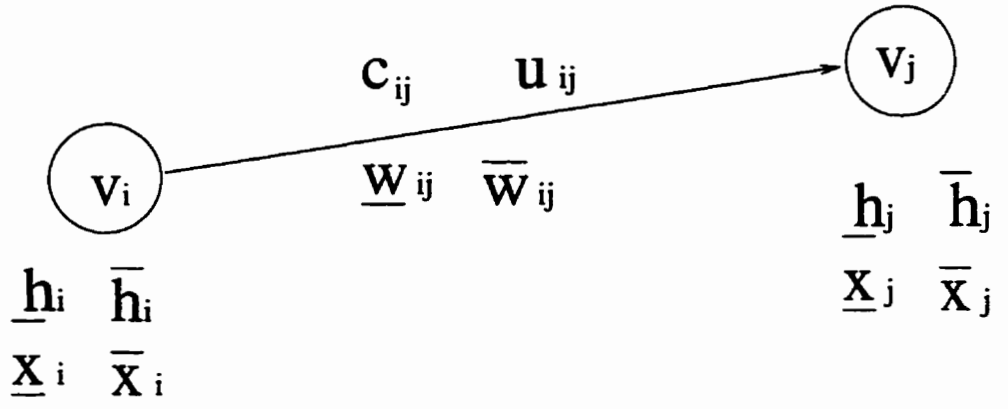


Figure 1.3 – *Notations*

Deux vecteurs, $\underline{h}_i = (\underline{h}_{i1}, \underline{h}_{i2}, ..., \underline{h}_{i|\mathcal{R}|})$ et $\overline{h}_i = (\overline{h}_{i1}, \overline{h}_{i2}, ..., \overline{h}_{i|\mathcal{R}|})$, correspondant respectivement aux seuils inférieurs et supérieurs sur l'accumulation des ressources, sont définis à chaque sommet. Des vecteurs de mise à jour, $\underline{x}_i = (\underline{x}_{i1}, \underline{x}_{i2}, ..., \underline{x}_{i|\mathcal{R}|})$ et $\overline{x}_i = (\overline{x}_{i1}, \overline{x}_{i2}, ..., \overline{x}_{i|\mathcal{R}|})$, associés respectivement aux seuils inférieurs et supérieurs, sont également donnés. Un coût c_{ij}, un vecteur d'utilisation de ressource $u_{ij} = (u_{ij1}, u_{ij2}, ..., u_{ij|\mathcal{R}|})$, ainsi que des vecteurs de bornes inférieures et supérieures $\underline{w}_{ij} = (\underline{w}_{ij1}, \underline{w}_{ij2}, ..., \underline{w}_{ij|\mathcal{R}|})$ et $\overline{w}_{ij} = (\overline{w}_{ij1}, \overline{w}_{ij2}, ..., \overline{w}_{ij|\mathcal{R}|})$ respectivement, sont aussi associés à chaque arc $(v_i, v_j) \in A$ (voir la figure 1.3).

Un ensemble d'arcs consécutifs du sommet v_1 à un sommet v_i sera appelé chemin v_1-v_i, comme dans les sections précédentes. Un tel chemin est caractérisé par un vecteur de ressource initial x_0, un vecteur de ressource courant $x_i = (x_{i1}, x_{i2}, ..., x_{i|\mathcal{R}|})$ et un coût c_i. Un chemin v_1-v_i peut être prolongé en utilisant un arc $(v_i, v_j) \in A$, uniquement si les contraintes de fenêtres ci-après sont satisfaites :

$$\underline{w}_{ijr} \leq x_{ir} \leq \overline{w}_{ijr}, \quad r \in \mathcal{R}. \tag{1.3}$$

Ainsi, pour un arc (v_i, v_j), les contraintes de fenêtres de ressource (1.3) définissent un intervalle de réalisabilité $[\underline{w}_{ijr}, \overline{w}_{ijr}]$ pour chaque ressource $r \in \mathcal{R}$. On considère, en outre, un opérateur de mise à jour $\Psi(\cdot,\cdot)$ tel que, pour tout sommet $v_j \in V$, toute ressource $r \in \mathcal{R}$ et pour tout vecteur de ressource $z \in \mathbf{Z}^{|\mathcal{R}|}$:

$$\Psi(v_j, z_r) = \begin{cases} \underline{x}_{jr} & \text{si } z_r < \underline{h}_{jr}, \\ z_r & \text{si } \underline{h}_{jr} \leq z_r \leq \overline{h}_{jr}, \\ \overline{x}_{jr} & \text{si } z_r > \overline{h}_{jr}. \end{cases}$$

Après la traversée de l'arc $(v_i, v_j) \in A$, c_{ij} et u_{ijr} sont respectivement ajoutés à c_i et à x_{ir}, pour $r \in \mathcal{R}$. L'opérateur de mise à jour $\Psi(\cdot,\cdot)$, est ensuite appliqué à $z_r = x_{ir} + u_{ijr}$ au sommet v_j pour obtenir $x_{jr} = \Psi(v_j, x_{ir} + u_{ijr})$, où $r \in \mathcal{R}$. La consommation de ressources à la source est donnée par $x_{1r} = \Psi(v_1, x_{0r})$, $r \in \mathcal{R}$. Dans la suite du texte, l'opérateur de mise à jour $\Psi(\cdot,\cdot)$ sera étendu à des arguments vectoriels, i.e., étant donné un sommet v_i et un vecteur de consommations de ressource $z \in \mathbf{Z}^{|\mathcal{R}|}$, le vecteur $(\Psi(v_i, z_1), \Psi(v_i, z_2), ..., \Psi(v_i, z_{|\mathcal{R}|}))$ sera noté $\Psi(v_i, z)$.

Un chemin v_1-v_n est réalisable si les contraintes de fenêtres de ressource (1.3) sont satisfaites sur chacun de ses arcs. Le nouveau problème de plus court chemin avec fenêtres de ressource se définit donc comme suit: étant donné un vecteur de ressource initial x_0, trouver un chemin v_1-v_n réalisable et de coût minimal, ou montrer qu'un tel chemin n'existe pas. Ce problème sera noté RCSPP. Il n'est pas difficile de voir que le problème RCSPP contient celui du sac à dos comme cas particulier et est, par conséquent, NP-difficile.

1.3.2 Un cas particulier

Le problème de plus court chemin avec fenêtres de ressource aux sommets discuté dans Desrosiers, Dumas, Solomon et Soumis [22] peut être considéré, dans le cas des graphes acycliques, comme un cas particulier, du problème RCSPP. En effet, soit un graphe orienté acyclique $G = (V, A)$ avec un ensemble de ressources

$\mathcal{R}$ à valeurs discrètes et, pour chaque arc $(v_i, v_j) \in A$, un coût c_{ij} et un vecteur de consommations de ressource u_{ij} comme précédemment. Des fenêtres de ressource $[a_{ir},\ b_{ir}]$, $r \in \mathcal{R}$, sont définies à chaque sommet $v_i \in V$. Un coût c_i et un vecteur de ressource $x_i = (x_{i1}, x_{i2}, ..., x_{i|\mathcal{R}|})$ sont associés à chaque chemin v_1-v_i.

Étant donné un vecteur de ressource initial x_0, en posant $x_1 = x_0$ et $c_1 = 0$, le problème de plus court chemin avec fenêtres de ressource aux sommets peut être défini comme celui de trouver un chemin v_1-v_n de coût minimum contenant uniquement des arcs $(v_i, v_j) \in A$ tels que:

$$a_{1r} \leq x_{1r} \leq b_{1r}, \quad c_j = c_i + c_{ij}, \quad x_{ir} + u_{ijr} \leq b_{jr}, \quad x_{jr} = \max\{a_{jr},\ x_{ir} + u_{ijr}\}, \quad r \in \mathcal{R}.$$

Par conséquent, en supprimant les fenêtres de ressource aux sommets et en posant:

$$\underline{h}_{ir} = \underline{x}_{ir} = a_{ir}, \quad \overline{h}_{ir} = \overline{x}_{ir} = b_{ir} \quad r \in \mathcal{R}, \quad v_i \in V,$$

$$\underline{w}_{ijr} = a_{ir}, \quad \overline{w}_{ijr} = b_{jr} - u_{ijr}, \quad r \in \mathcal{R}, \quad (v_i, v_j) \in A;$$

on obtient une instance de RCSPP.

Il convient de remarquer que les bornes inférieures des fenêtres dans le problème de plus court chemin avec fenêtres de ressource aux sommets sont *molles*, i.e., un chemin v_1-v_n optimal peut contenir des arcs $(v_i, v_j) \in A$ tels que $x_{ir} + u_{ijr} \leq a_{jr}$. On peut voir aisément qu'une version simplifiée de RCSPP où les seuils sont égaux aux valeurs de mise à jour correspondantes, peut se transformer en une instance du problème de plus court chemin avec fenêtres de ressource aux sommets. Il suffit, principalement, d'introduire un sommet intermédiaire sur chaque arc et de dédoubler les ressources impliquant des contraintes dures pour les fenêtres (voir, par exemple, Gamache *et al.* [32]). Ce dédoublement de ressources constitue cependant un inconvénient qui rend cette transformation inverse peut attrayante.

Différents algorithmes sont présentés au chapitre 4 pour le problème RCSPP, de même qu'une discussion de leurs complexités. En particulier, une procédure est décrite pour les cas de réoptimisation, lorsque le problème de plus court chemin avec contraintes apparaît comme un problème auxiliaire de génération de colonnes. Nous présentons dans la prochaine section un exemple d'application nécessitant une telle

résolution répétitive d'un problème complexe de cheminement dans un réseau, qui se ramène au problème RCSPP.

1.4 Le problème d'horaires de personnel soignant

Le problème de confection d'horaires de personnel soignant consiste à générer une configuration d'horaires individuels, i.e, de séquences d'affectations journalières sur un certain horizon de planification. La configuration d'horaires est générée de manière à satisfaire les spécifications de la convention collective et les quotas de demande exprimés par l'hôpital, tout en minimisant le coût salarial et en maximisant les préférences individuelles des infirmières ainsi que la qualité des soins.

Les spécifications de la convention collective sont des règles qui permettent de définir des horaires acceptables pour chaque infirmière individuellement, en termes d'ancienneté, de charge de travail, de congés statutaires ou de fins de semaines, et d'affectations consécutives, incluant les rotations entre divers types de quarts. Une affectation est une spécification du quart de travail qu'une personne doit effectuer un jour donné. Un quart de travail est soit de jour, du soir ou de nuit. Il est caractérisé par une heure de début et une heure de fin qui sont fixes.

Un jour est divisé en plusieurs périodes de demande caractérisées par des heures fixes de début et de fin. Les périodes de demande sont identiques pour tous les jours mais ne coïncident pas nécessairement avec les quarts de travail. Un quart de travail peut éventuellement couvrir plusieurs périodes de demande. On supposera que les périodes de demande originales sont décomposées, si nécessaire, en plusieurs périodes plus petites, de sorte qu'un quart de travail couvre une période de demande entièrement ou pas du tout (voir la figure 1.4). Dans la suite du texte, l'expression "période de demande" désignera ces périodes modifiées, sauf indication contraire.

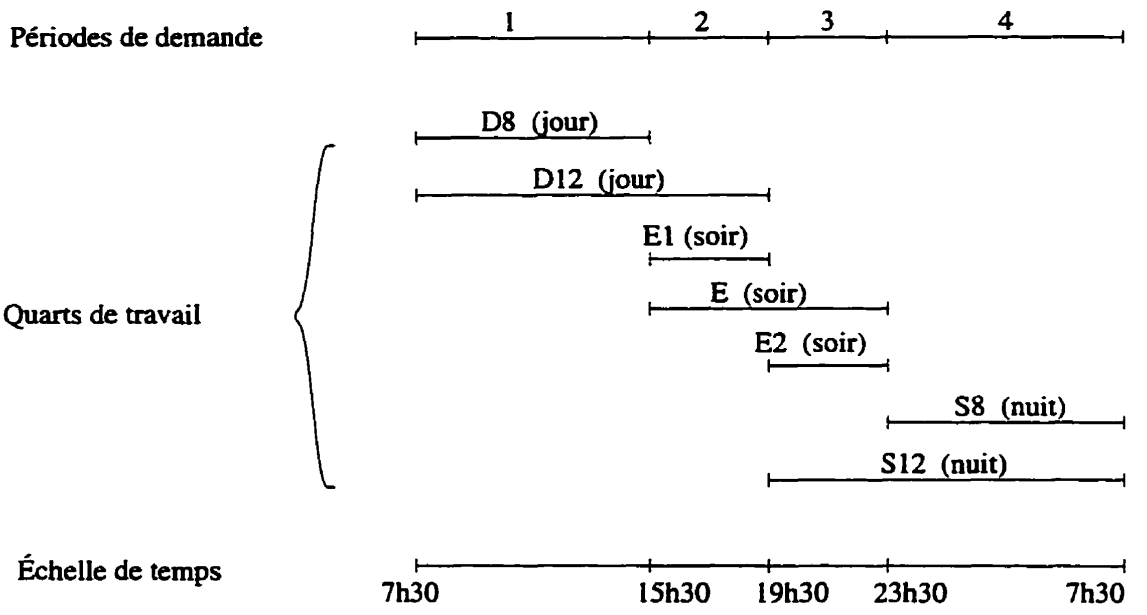


Figure 1.4 – *Exemple de quarts de travail et de périodes de demande*

Les quotas de demandes spécifient, pour chacune des périodes de demande et pour chaque jour, le nombre de personnes de chaque niveau de qualification, ou combinaison de niveaux de qualification, qui doivent être présentes. Des spécifications relatives aux quarts de travail peuvent également être incluses dans l'expression des quotas de demande.

Le coût salarial comprend le salaire régulier du personnel permanent ainsi que les coûts des temps supplémentaires et celui du personnel flottant. Les préférences individuelles peuvent être exprimées en termes de requêtes pour des jours de congés, des quarts de jours par rapport à ceux de nuit, etc. La qualité des soins peut être évaluée par le niveau d'équilibre entre les personnes expérimentées et moins expérimentées qui sont affectées à la même période de demande.

Ce travail de modélisation du problème d'horaires de personnel soignant a été fait en collaboration avec l'Hôpital Royal Victoria de Montréal. Le but visé est de développer un système de confection d'horaires qui permettra de prendre en compte les besoins et les contraintes spécifiques à l'unité considérée, sans recourir à des horaires-types fournis par l'infirmière-chef. Le système ne devra pas demander

beaucoup de travail de mise à jour, étant donné que les besoins et les contraintes varient d'un horizon à l'autre. Une explication exhaustive des multiples règles à respecter et des compromis acceptables a été donnée par le personnel de l'hôpital.

La confection manuelle des horaires est très fastidieuse et se limite généralement à la recherche d'une solution réalisable, avec peu d'accent sur l'optimisation, du fait de la complexité des règles de la convention collective. La prise en compte de ces multiples contraintes est un défi de taille pour l'automatisation de la confection des horaires, comme l'illustre la revue de la littérature ci-après.

1.4.1 Une bibliographie des horaires d'infirmières

L'approche manuelle par essais et erreurs a été abondamment étudiée dans la littérature spécialisée en administration de la santé (voir, par exemple, la récente revue de la littérature présentée dans Hung [39]). Une grande partie des travaux sur l'automatisation de la confection des horaires de personnel soignant considère des systèmes sans modèles mathématiques explicites (exemple., Smith et Wiggins [60], Anzai et Miura [4], Okada et Okada [52]), qui sont essentiellement une traduction informatique de la logique de l'approche manuelle.

Plusieurs articles de la littérature traitent cependant de modèles cycliques, qui sont des modèles mathématiques assez simplifiés et susceptibles de résolution exacte. Dans ces modèles, des séquences d'affectations cycliques sont générées de manière à minimiser la taille du personnel nécessaire pour couvrir la demande, également supposée cyclique.

Baker [5] considère, par exemple, deux jours consécutifs de congé par semaine pour chaque personne, tandis que Burns [10] étudie le cas de dix jours de travail sur quatorze avec congé une fin de semaine sur deux et jusqu'à six jours de travail consécutifs. Burns et Koop [11] considèrent des affectations cycliques dans un modèle similaire, mais avec trois types de quarts de travail et des spécifications cycliques fixes pour la répartition des jours de travail et de congé.

Les premiers modèles réalistes et non cycliques sont proposés dans Warner [62] et dans Miller *et al.* [49]. Ces modèles sont cependant résolus de manière heuristique, essentiellement à cause de la très grande taille de l'ensemble des horaires potentiels pour chaque personne. Les modèles décrits traitent de la maximisation des préférences individuelles et comprennent des procédures pour tenir compte des requêtes personnelles.

Certains travaux, plus récents, considèrent des modèles multiobjectifs non cycliques, qui sont résolus au moyen de la programmation par buts (par exemple, Ozkarahan et Bailey [53]), de méthodes interactives de programmation multicritère ou de la méta-heuristique taboue avec une fonction d'utilité lexicographique (par exemple, Berrada [9]). Des fonctions de priorité et de performance sont définies pour ces modèles multiobjectifs à partir de certaines hypothèses sur la couverture des demandes, les objectifs de la direction et les préférences des infirmières.

On notera cependant que les méthodes de résolution utilisées par ces auteurs sont plutôt restrictives par rapport au nombre d'horaires possibles par personne. En outre, l'approche interactive de programmation mathématique multicritère décrite implique une trop grande intervention du décideur. Celui-ci doit, à chaque itération majeure, rajuster sélectivement les préférences pour permettre la génération d'une nouvelle solution non dominée.

1.4.2 Limitations des systèmes existants

La plupart des algorithmes exacts de la littérature considèrent des situations simplifiées (telles que des modèles cycliques) et sont, par conséquent, peu réalistes. Par ailleurs, d'importants problèmes de réalisabilité, sources de griefs de travail, peuvent se rencontrer dans les systèmes heuristiques. La solution obtenue, dans ce dernier cas, peut être systématiquement sous-optimale, sans aucune indication précise sur les améliorations encore possibles. La plupart des systèmes heuristiques sont également sensibles à l'environnement considéré et manquent ainsi de flexibilité en cas de modification de celui-ci.

Une observation attentive du problème d'horaires de personnel soignant indique que l'une des difficultés principales provient de la taille de l'ensemble des horaires potentiels. Si $\mathcal{D}$ est l'ensemble des jours de l'horizon et une infirmière peut effectuer T quarts de travail différents, alors $O(T^{|\mathcal{D}|})$ horaires doivent être implicitement examinée pour cette infirmière.

Cette remarque, ainsi que la perspective de capitaliser sur les avancées en programmation mathématique et sur les récents développements des équipements informatiques, motivent l'approche exacte de génération de colonnes proposée dans cette thèse. Nous présentons ci-après un survol du modèle et une brève revue de littérature sur la programmation linéaire généralisée.

1.4.3 Un survol du modèle de génération de colonnes

Le modèle considéré comporte un problème maître qui implique un objectif et des contraintes relatives à l'ensemble de la configuration des horaires générés. Le modèle comporte également un problème auxiliaire traitant des spécifications relatives à une infirmière donnée. Le problème maître est un programme linéaire en variables 0-1 qui détermine une configuration d'horaires pour satisfaire la demande tout en minimisant les coûts salariaux et en maximisant les préférences.

Chaque colonne dans la matrice des contraintes du problème correspond à un horaire réalisable pour une infirmière. Compte tenu de la taille de cette matrice, seul un petit nombre de colonnes sont considérées à la fois. D'autres colonnes sont générées au fur et à mesure qu'elles sont nécessaires pour améliorer la solution courante.

Cette recherche de nouvelles colonnes se fait en résolvant un problème auxiliaire de plus court chemin avec fenêtres de ressource. Un chemin réalisable dans le réseau associé correspond à un horaire acceptable pour l'infirmière considérée. La réalisabilité dans le problème auxiliaire est définie à partir des règles de la convention collective, telles qu'elles s'appliquent à l'infirmière. Cela conduit à une structure des contraintes de ressource qui correspond au modèle présenté à la section 1.3.

La résolution de tels programmes linéaires généralisés en variables entières (ou mixtes) comporte une phase de résolution de la relaxation linéaire du problème maître et une phase de recherche de solutions entières (ou mixtes). La génération de colonnes en programmation linéaire a été introduite par Dantzig et Wolfe [17], tandis qu'une première heuristique pour les programmes linéaires généralisés en variables entières a été présentée par Gilmore et Gomory [33, 34] dans une une étude sur le problème de découpe.

Plusieurs auteurs ont, par la suite, combiné la méthode d'évaluation et de séparation progressives avec la génération de colonnes pour résoudre des problèmes de grande taille dans divers domaines d'applications. Appelgren [1] et Levine [45] sont parmi les premiers auteurs à étudier des modèles de programmation linéaire généralisée avec fenêtres de temps. La plupart des travaux de cette catégorie sont couverts dans la récente revue bibliographique de Desrosiers *et al.* [22] (voir aussi Desaulniers *et al.* [18]) sur les problèmes de routage et de distribution.

La résolution optimale de ces problèmes comporte des décisions de branchements qui doivent être compatibles avec les structures respectives de la relaxation linéaire du problème maître et du problème auxiliaire de génération de colonnes. Lorsqu'une variable fractionnaire d'une base optimale est fixée à zéro, une précaution particulière doit être prise pour éviter de regénérer une colonne lui correspondant. Plusieurs algorithmes (par exemple, Hansen *et al.* [37], Desrosiers *et al.* [22], Vanderbeck et Wolsey [61], Desaulniers *et al.* [18], voir aussi Barnhart *et al.* [6] pour une revue bibliographique) exposent différentes manières pour obtenir une solution entière optimale.

Le modèle présenté dans cette thèse diffère de ceux traités dans Desrosiers *et al.* [22], par la structure des fenêtres de ressource dans le problème auxiliaire de plus court chemin. La spécificité des règles de la convention collective se traduit par des contraintes de mise à jour des ressources qui correspondent au modèle présenté à la section 1.3. Les détails de la modélisation et de la résolution de ce problème d'horaires sont décrits au chapitre 5.

CHAPITRE 2

ALGORITHMES DE CHEMINS AVEC ÉTENDUE OU RATIO MINIMUM

2.1 Problèmes à un critère

Nous décrivons, ci-dessous, un algorithme, dénommé MRG, qui permet de résoudre le problème MINRANGE introduit à la section 1.1. Étant données la meilleure étendue connue R_{opt} et une valeur $\bar{c}$, l'algorithme MRG calcule la plus grande valeur $\underline{c}$ telle qu'il existe un chemin dont aucun arc n'a une longueur plus petite que $\underline{c}$ ni plus grande que $\bar{c}$, en résolvant un problème MAXMIN. Un chemin d'étendue minimale ayant ces caractéristiques et dont l'étendue est plus petite que R_{opt}, est ensuite déterminé. Cela se fait en résolvant un problème MINMAX par le biais d'une procédure pouvant éventuellement se terminer avec la preuve qu'un tel chemin n'existe pas.

La longueur $\tilde{c}$, de l'arc le plus long sur le chemin ainsi trouvé, est utilisée pour changer la valeur de $\bar{c}$ à $(\underline{c} - 1) + (\bar{c} - \underline{c}) - 1$. Cette mise à jour se justifie par le fait qu'une décroissance de $\bar{c}$ entraîne une plus petite valeur de $\underline{c}$ et, ainsi, une étendue R_{opt} plus petite sera cherchée à l'itération suivante. Le processus commence avec une valeur de $\bar{c}$ qui est une borne supérieure sur toutes les longueurs d'arcs dans le graphe. Il est ensuite répété pour des valeurs décroissantes de $\bar{c}$ jusqu'à ce qu'aucun nouveau chemin ne puisse plus être trouvé.

Algorithme MRG (MINRANGE)

a) *Initialisation.*
Poser $R_{opt} = \max_{(v_i,v_j)\in A} c_{ij} - \min_{(v_i,v_j)\in A} c_{ij} + 1$ et $\bar{c} = \max_{(v_i,v_j)\in A} c_{ij}$.
(R_{opt} est la valeur du meilleur chemin connu P_{opt}).

b) *Maximisation de la longueur d'arc minimale.*
Poser $\bar{G} = (V, \bar{A})$ où $\bar{A} = \{(v_i, v_j) \in A | c_{ij} \leq \bar{c}\}$.
Soit $\mathcal{P}(\bar{G})$ la restriction de $\mathcal{P}$ à $\bar{G}$. Résoudre le sous-problème:

$$\max_{P\in\mathcal{P}(\bar{G})} \min_{(v_i,v_j)\in P} c_{ij}. \tag{2.1}$$

Soit $\underline{c}$ la valeur optimale de (2.1). Si $\mathcal{P}(\bar{G}) = \emptyset$, FIN, R_{opt} est la valeur optimale du problème MINRANGE et P_{opt} le chemin optimal.

c) *Minimisation de la longueur d'arc maximale.*
Poser $\tilde{G} = (V, \tilde{A})$ où $\tilde{A} = \{(v_i, v_j) \in A | \underline{c} \leq c_{ij} \leq min(\bar{c}, \underline{c} + R_{opt} - 1)\}$.
Soit $\mathcal{P}(\tilde{G})$ la restriction de $\mathcal{P}$ à $\tilde{G}$. Résoudre le sous-problème:

$$\min_{P\in\mathcal{P}(\tilde{G})} \max_{(v_i,v_j)\in P} c_{ij} \quad . \tag{2.2}$$

Soit $\tilde{c}$ la valeur optimale de (2.2) et $\tilde{P}$ le chemin correspondant.
Si $\mathcal{P}(\tilde{G}) = \emptyset$, poser $\tilde{c} = \max_{(v_i,v_j)\in A} c_{ij} + 1$.

d) *Mise à jour des paramètres.*

Si $\tilde{c} - \underline{c} < R_{opt}$ poser $R_{opt} = \tilde{c} - \underline{c}$, $P_{opt} = \tilde{P}$.
Poser $\bar{c} = \underline{c} + R_{opt} - 2$ et retourner à l'étape b).

On peut aisément modifier l'algorithme MRG pour obtenir un algorithme pour le problème MINRATIO. Cependant, les deux algorithmes restent fondamentalement les mêmes. Dans l'algorithme MRT ci-dessous, R_{opt} représente le meilleur ratio connu. La valeur de $\underline{c}$ est calculée comme dans l'algorithme MRG, mais le problème MIN-MAX est résolu sur un sous-graphe où aucun chemin ayant un ratio égal ou supérieur à R_{opt} ne peut être trouvé.

La valeur de $\bar{c}$ est ensuite posée égale à $\lceil(\underline{c}-1)\times R_{opt}\rceil - 1$, exploitant ainsi le fait qu'une diminution de $\bar{c}$ produit une plus petite valeur de $\underline{c}$ et que les ratios égaux ou supérieurs à R_{opt} peuvent également être sautés. En effet, la solution suivante du problème MAXMIN sera inférieure ou égale à $\underline{c}-1$ et la plus grande longueur d'arc de la prochaine solution du problème MINMAX doit donc être plus petite ou égale à $(\underline{c}-1)\times R_{opt}$.

Algorithme MRT (MINRATIO)

Appliquer l'algorithme MRG avec les modifications suivantes, dues au changement de fonction objectif:

- À l'étape a) poser
$$R_{opt} = \frac{\max_{(v_i,v_j)\in A} c_{ij}}{\min_{(v_i,v_j)\in A} c_{ij}} + 1.$$
À l'étape c) poser
$$\bar{A} = \{(v_i,v_j)\in A | \underline{c} \leq c_{ij} \leq min(\bar{c}, \lceil \underline{c}\times R_{opt}\rceil - 1)\}$$
où $\lceil a\rceil$ est le plus petit entier non inférieur à a.
- Remplacer l'étape d) par:
Si $\frac{\bar{c}}{\underline{c}} < R_{opt}$ poser $R_{opt} = \frac{\bar{c}}{\underline{c}}$, $P_{opt} = \bar{P}$.
Poser $\bar{c} = \lceil(\underline{c}-1)\times R_{opt}\rceil - 1$ et retourner à l'étape b).

Les algorithmes MRG et MRT sont illustrés sur le petit exemple de la figure 2.1 et les détails de la résolution sont donnés aux tableaux 2.1 et 2.2 respectivement. On remarquera que (v_1,v_4,v_8), (v_1,v_4,v_6,v_7,v_8) et (v_1,v_3,v_5,v_6,v_8) sont tous des chemins efficaces pour le problème bicritère MINMAX-MAXMIN, avec comme vecteurs objectifs, $(3,2)$, $(5,3)$ et $(8,6)$ respectivement. Cependant, les deux algorithmes, MRG et MRT, sautent le chemin (v_1,v_4,v_6,v_7,v_8). Nous justifions ci-après les algorithmes et donnons leur complexité.

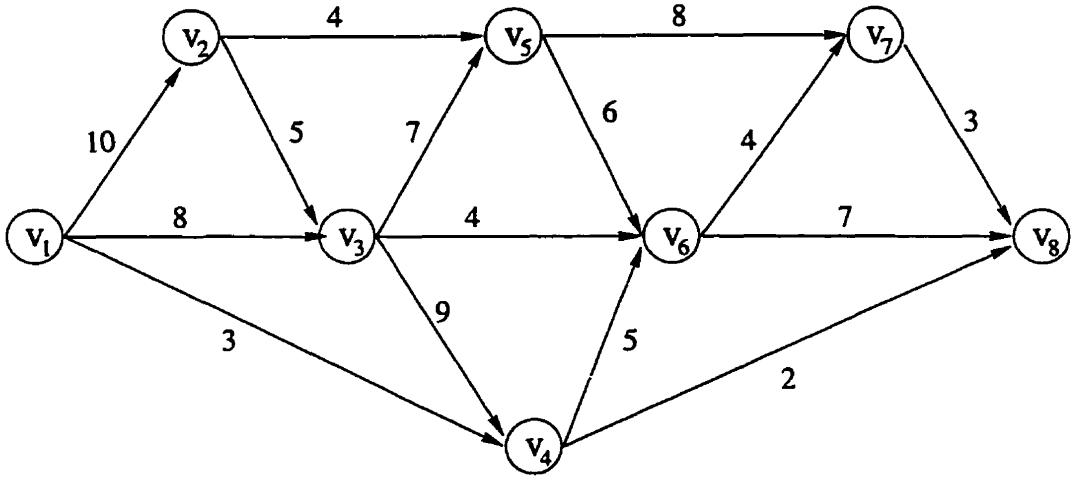


Figure 2.1 – *Exemple pour les algorithmes MRG et MRT*

Théorème 2.1 *L'algorithme MRG (MRT) résout le problème MINRANGE (MINRATIO) en temps $O(m^2 \log n)$ sur un graphe orienté et en temps $O(m^2)$ sur un graphe non orienté.*

Preuve. Examinons d'abord l'exactitude de l'algorithme MRG. Il y au plus $\frac{m(m-1)}{2}$ valeurs possibles pour l'étendue, $\bar{c} - \underline{c}$, des longueurs d'arcs le long d'un chemin v_1-v_n, i.e., autant que de paires d'arcs dans le graphe G. Comme indiqué à la section 1.1.1 le chemin optimal P_{opt} doit être efficace pour le problème bicritère MINMAX-MAXMIN. Ainsi, on pourrait considérer toutes les $O(m)$ valeurs possibles de $\bar{c}$ et trouver les valeurs correspondantes de $\underline{c}$. Cependant, plusieurs valeurs possibles sont implicitement prises en compte. En effet, pour une valeur donnée de $\bar{c}$, on trouve, à l'étape b), la valeur correspondante de $\underline{c}$ et on détermine, à l'étape c), la meilleure valeur de $\bar{c}$ associée à $\underline{c}$. Toutes les valeurs entre $\bar{c}$ et $\bar{c}+1$ correspondent alors à des chemins non efficaces et sont traités implicitement.

En outre, la valeur R_{opt} du meilleur chemin connu est utilisée de deux façons pour éliminer d'autres chemins. D'abord, à l'étape c), on remarque qu'une amélioration de R_{opt} ne peut avoir lieu que si $\bar{c} \leq \underline{c} + R_{opt} - 1$ et les arcs ayant une longueur plus grande que $\bar{c}$ ne sont pas considérés dans $\tilde{G}$. Ensuite, à l'étape d), on observe qu'étant donné que le meilleur chemin pour la valeur de $\underline{c}$ est connu, on doit avoir une diminution de la valeur de $\underline{c}$ pour le prochain chemin à considérer. Un meilleur

Tableau 2.1 - *Illustration de l'algorithme MRG*

Itér.	Étape a)	Étape b)	Étape c)	Étape d)
1	$P_{opt} = (-)$	$\underline{c} = 6$	$\bar{c} = 8$	$R_{opt} = 2$, $\bar{c} = 6$
	$R_{opt} = 9$, $\bar{c} = 10$	$\mathcal{P}(\overline{G}) \neq \emptyset$	$\tilde{p} = (v_1, v_3, v_5, v_6, v_8)$	$p_{opt} = (v_1, v_3, v_5, v_6, v_8)$
2		$\underline{c} = 3$	$\mathcal{P}(\tilde{G}) = \emptyset$	$\bar{c} = 3$
		$\mathcal{P}(\overline{G}) \neq \emptyset$	$\tilde{c} = 11$	
3		$\underline{c} = 2$	$\bar{c} = 3$	$R_{opt} = 1$, $\bar{c} = 1$
		$\mathcal{P}(\overline{G}) \neq \emptyset$	$\tilde{p} = (v_1, v_4, v_8)$	$P_{opt} = (v_1, v_4, v_8)$
4		$\mathcal{P}(\overline{G}) = \emptyset$		
		Fin		

chemin ne peut être obtenu que si $\bar{c} \leq \underline{c} - 1 + R_{opt} - 1$, i.e., $\bar{c} \leq \underline{c} + R_{opt} - 2$. Ainsi donc, tous les chemins, efficaces ou non, correspondant à R_{opt} sont évités dans la prochaine itération. L'exactitude de l'algorithme MRT peut être prouvée par des arguments similaires.

En ce qui concerne la complexité, on notera que l'étape a) est en $O(m)$ et l'étape d) en $O(n)$. Les étapes cruciales sont b) et c). Chacune d'elles se ramène à un problème de capacité maximale dans $\tilde{G}$ avec des capacités $\bar{c} - c_{ij}$ et dans $\tilde{G}$ avec des capacités $c_{ij} - \underline{c}$ respectivement. Dans un graphe orienté, ce problème peut se résoudre par une variante de l'algorithme de Dijkstra [25]. Il suffit, pour cela, d'étiqueter un sommet courant v_i avec la capacité minimale sur un chemin v_1-v_i, de capacité maximale, passant uniquement par des sommets étiquetés et déjà sélectionés, et de sélectionner itérativement le sommet de capacité maximale.

Ainsi, les étapes b) et c) requièrent un temps $O(m \log n)$ quand G est orienté, tout comme les algorithmes MRG et MRT. Si G est non orienté, l' algorithme en $O(m)$ de Punnen [54], qui a la meilleure complexité possible pour le problème du chemin de capacité maximale, peut être utilisé et la complexité des algorithmes MRG et MRT se réduit à $O(m^2)$. □

Tableau 2.2 – *Illustration de l'algorithme MRT*

Itér.	Étape a)	Étape b)	Étape c)	Étape d)
1	$P_{opt} = (-)$	$\underline{c} = 6$	$\bar{c} = 8$	$R_{opt} = \frac{4}{3}$, $\overline{c} = 6$
	$R_{opt} = 6$, $\overline{c} = 10$	$\mathcal{P}(\overline{G}) \neq \emptyset$	$\tilde{p} = (v_1, v_3, v_5, v_6, v_8)$	$P_{opt} = (v_1, v_3, v_5, v_6, v_8)$
2		$\underline{c} = 3$	$\mathcal{P}(\bar{G}) = \emptyset$	$\overline{c} = 2$
		$\mathcal{P}(\overline{G}) \neq \emptyset$	$\bar{c} = 11$	
3		$\mathcal{P}(\overline{G}) = \emptyset$		
		Fin		

Remarquons qu'il existe toujours une solution au problème MINRANGE ou MINRATIO, correspondant à un chemin optimal élémentaire, i.e., ne passant pas deux fois par le même sommet. Ce n'est pas nécessairement le cas si l'on considère la maximisation de l'étendue ou du ratio plutôt que la minimisation.

2.2 Algorithmes bicritères

Nous nous intéressons, dans cette section, à la résolution des extensions bicritères MINRANGE-MAXMIN et MINRANGE-MINSUM discutés à la section 1.1.1. Rappelons que le chemin optimal pour le critère MINRANGE appartient à l'ensemble des chemins efficaces pour le problème bicritère MINMAX-MAXMIN. Celui-ci peut être résolu en temps $O(m^2 \log n)$ dans un graphe orienté par un algorithme décrit dans Hansen [36] et en temps $O(m^2)$ dans le cas non orienté par l'algorithme de Punnen [54]. L'utilisation de ces algorithmes dans la procédure ci-après permet aisément de résoudre les problèmes bicritères MINRANGE-MAXMIN et MINRANGE-MINMAX ou leurs variantes avec le critère du ratio.

Algorithme MRGMM (MINRANGE-MINMAX)

a) *Chemins efficaces*
Trouver un ensemble complet de chemins efficaces non équivalents dans G pour les critères MINMAX et MAXMIN.

b) *Valeurs*
Calculer les valeurs de l'étendue pour tous les chemins efficaces trouvés en a).

c) *Classement et suppression*
Classer tous les chemins efficaces trouvés en a) par ordre non décroissant des valeurs de l'étendue et, en cas d'ex-aequo, de la valeur $\bar{c}$ du critère MINMAX. Supprimer les chemins dominés.

L'étape dominante dans cette procédure est a), à la fois pour les graphes orientés et non orientés. Ainsi, le temps requis est $O(m^2 \log n)$ et $O(m^2)$ respectivement. L'algorithme MRGMM est appliqué au graphe de la figure 2.2 et les détails de la résolution sont résumés dans le tableau 2.3.

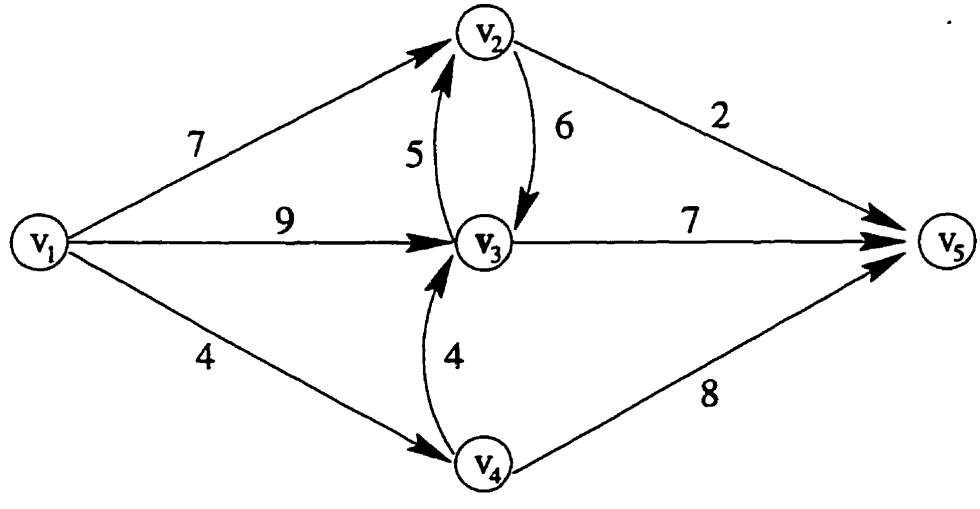


Figure 2.2 – *Exemple pour les algorithmes MRGMM et MRGMS*

Tableau 2.3 – *Illustration de l'algorithme MRGMM*

Étapes	Chemins	Critère Maxmin	Critère Minmax	Étendue	Solutions efficaces
a)	$(v_1, v_4, v_3, v_2, v_5)$	2	6		
	(v_1, v_2, v_3, v_5)	5	7		
	(v_1, v_3, v_5)	7	9		
b)	$(v_1, v_4, v_3, v_2, v_5)$			4	
	(v_1, v_2, v_3, v_5)			2	
	(v_1, v_3, v_5)			2	
c)	(v_1, v_2, v_3, v_5)		7	2	√
	(v_1, v_3, v_5)		9	2	
	$(v_1, v_4, v_3, v_2, v_5)$		6	4	√

Nous abordons maintenant la présentation d'un algorithme, noté MRGMS, pour résoudre le problème bicritère MINRANGE-MINSUM, i.e., pour déterminer un ensemble complet de chemins efficaces non équivalents pour les critères de l'étendue et de la longueur. L'algorithme effectue des tests de dominance pour un ensemble de chemins candidats . Ceux-ci sont générés en calculant itérativement des plus courts chemins dont les longueurs d'arcs sont comprises entre des bornes inférieures $\underline{c}$ et supérieures $\bar{c}$ mobiles.

Étant donnée une valeur de $\bar{c}$, des plus courts chemins sont calculés pour des valeurs consécutives décroissantes de $\underline{c}$ jusqu'à ce que $\bar{c} - \underline{c}$ soit égale à l'étendue de la solution optimale pour le critère de la longueur du chemin (qui est l'étendue maximale pour tout chemin efficace). Ensuite, $\bar{c}$ est mis à jour en le posant égal à la plus grande longueur d'arc plus petite que la valeur courante de $\bar{c}$, de manière à ne considérer qu'une seule fois les valeurs identiques de $\bar{c}$.

Différents tests d'accélération sont utilisés pour sauter certaines valeurs de $\underline{c}$. En particulier, une paire $(\underline{c}, \bar{c})$ est sautée si une autre paire $(\underline{c}', \bar{c}')$, telle que $\underline{c}' \leq \underline{c}$ et $\bar{c}' \geq \bar{c}$, a déjà été examinée et a conduit à un plus court chemin dont la longueur n'est pas plus petite que celle du chemin optimal pour le critère de l'étendue (qui est la

longueur maximale pour un chemin efficace). Le couple $(\underline{c}, \bar{c})$ est également rejeté si $\underline{c}' = \underline{c}$ et $\bar{c}' \geq \bar{c}$, et la plus grande longueur d'arc du plus court chemin correspondant est plus petite ou égale à $\bar{c}$, puisque, autrement, le même chemin serait recalculé.

Dans l'algorithme MRGMS, $\ell(\underline{c})$ représente la longueur du plus court chemin calculé à l'étape f), la dernière fois qu'un intervalle ayant $\underline{c}$ comme borne inférieure a été considéré. La valeur de $r(\underline{c})$ définit une borne supérieure sur la plus grande longueur d'arc de ce chemin. L_{opt} est la longueur du dernier plus court chemin déterminé pour la valeur courante de $\bar{c}$. La dernière valeur de $\underline{c}$ pour laquelle la longueur du plus court chemin est plus grande ou égale à la longueur maximale de tout chemin efficace, est notée $\underline{c}^*$. La plus petite valeur de $\underline{c}$ pour laquelle un plus court chemin a été trouvé est représentée par $\underline{c}^{**}$.

Algorithme MRGMS (MINRANGE-MINSUM)

a) *Chemin efficace de longueur minimale*
Trouver un plus court chemin v_1-v_n dans G et poser L^1_{opt} égal à sa longueur . Soit $\hat{G}$ le sous-graphe des plus courts chemins v_1-v_n dans G, trouver un chemin P^1 d'étendue minimale dans $\hat{G}$, et poser son étendue égale à R^1_{opt}. Garder $(P^1, L^1_{opt}, R^1_{opt})$ dans une liste $\mathcal{L}$ de chemins efficaces (candidats).

b) *Chemin efficace d'étendue minimale*
Trouver toutes les paires de longueurs d'arcs, $\bar{c}$, $\underline{c}$, pour lesquels $\bar{c} - \underline{c}$ est l'étendue minimale R^2_{opt} des chemins v_1-v_n. Pour chacune de ces paires, considérer le sous-graphe $G^* = (V, A^*)$ où $A^* = \{(v_i, v_j) \in A | \underline{c} \leq c_{ij} \leq \bar{c}\}$ et trouver un plus court chemin P^2, de longueur L^2_{opt}, dans G^*. Garder $(P^2, L^2_{opt}, R^2_{opt})$ dans $\mathcal{L}$.

c) *Classement des arcs*
Classer tous les arcs de G par ordre non croissant de leurs longueurs (en départageant arbitrairement les ex-aequos). Soient c_i, pour $i = 1, 2, ..., m$, les longueurs d'arcs après classement, poser $\ell(c_i) = +\infty$, $r(c_i) = c_1$ ainsi que $c_{m+1} = c_m - 1$, $\bar{c} = c_1$, $\underline{c}^* = \underline{c}^{**} = c_1 + 1$, $L_{opt} = +\infty$, et $k = 1$.

d) *Maximisation de la longueur d'arc minimale*
Poser $\bar{G} = (V, \bar{A})$ où $\bar{A} = \{(v_i, v_j) \in A | c_{ij} \leq \bar{c}\}$ et $\mathcal{P}(\bar{G})$ est la restriction de $\mathcal{P}$ à $\bar{G}$. Résoudre le sous-problème:

$$\max_{P \in \mathcal{P}(\bar{G})} \min_{(v_i, v_j) \in P} c_{ij}. \tag{2.3}$$

Soit $\underline{c}$ la valeur optimale de (2.3) et j l'indice d'arc tel que $j = \min\{i | c_i = \underline{c}\}$; si $\mathcal{P}(\bar{G}) = \emptyset$, aller à la dernière étape.

e) *Tests d'étendue et de dominance*
Si $\bar{c} - \underline{c} \geq R^1_{opt}$ ou $\underline{c} = c_{m+1}$, faire $k \leftarrow \min\{i | c_i < c_k\}$, poser $\bar{c} = c_k$, $L_{opt} = +\infty$ et retourner à d). Si $\underline{c} \geq \underline{c}^*$, faire $j \leftarrow \min\{i | c_i < \underline{c}^*\}$, poser $L_{opt} = \ell(\underline{c}^*)$, $\underline{c} = c_j$ et répéter l'étape courante. Si $\underline{c} \geq \underline{c}^{**}$ et $r(\underline{c}) \leq \bar{c}$, faire $j \leftarrow \min\{i | c_i < c_j\}$, poser $L_{opt} = \ell(\underline{c})$, $\underline{c} = c_j$ et répéter l'étape courante.

f) *Minimisation de la longueur du chemin*
Déterminer un plus court chemin P^*, de longueur $L(P^*)$ et d'étendue $R(P^*)$, dans G^*. Poser $\ell(\underline{c}) = L(P^*)$. Si $L(P^*) < L_{opt}$, poser $r(\underline{c}) = \underline{c} + R(P^*)$, sinon $r(\underline{c}) = \bar{c}$. Si $L(P^*) \geq L^2_{opt}$, poser $\underline{c}^* = \underline{c}$. Si $L(P^*) < L^2_{opt}$, garder $(P^*, L(P^*), R(P^*))$ dans $\mathcal{L}$. Faire $\underline{c}^{**} \leftarrow \min\{\underline{c}^{**}, \underline{c}\}$ et $j \leftarrow \min\{i | c_i < c_j\}$, puis poser $L_{opt} = \ell(\underline{c})$, $\underline{c} = c_j$ et retourner à e).

g) *Suppression des chemins dominés*
Classer les chemins P de $\mathcal{L}$ par valeurs non décroissantes de $R(P)$ et de $L(P)$ en cas d'ex-aequo. Si $L(P) \geq L(P')$ où P' est le prédécesseur de P dans la liste ordonnée $\mathcal{L}$, supprimer P. FIN: $\mathcal{L}$ contient un ensemble complet de chemins efficaces non équivalents.

Théorème 2.2 *L'algorithme MRGMS détermine un ensemble complet de chemins efficaces non équivalents pour les critères de l'étendue et de la longueur en temps* $O(m^3 \log n)$.

Preuve. L'algorithme considère implicitement toutes les instances possibles du couple formé par la longueur d'arc maximale $\bar{c}$ et la longueur d'arc minimale $\underline{c}$ sur un chemin. Des valeurs de $\bar{c}$ sont sautées à l'étape e) seulement en cas d'ex-aequos. Pour une

valeur fixée de $\bar{c}$, des valeurs de $\underline{c}$ sont sautées aux étapes e) et f) en cas d'ex-aequo et également à l'étape e) dans les trois cas ci-après.

D'abord, si l'étendue $\bar{c}-\underline{c}$ est plus grande ou égale à l'étendue maximale R^1_{opt} de tout chemin efficace. Ensuite, si $[\underline{c},\ \bar{c}]$ est contenu dans un intervalle déjà considéré, allant d'une valeur $\underline{c}^*$ plus petite ou égale à $\underline{c}$, à une valeur plus grande que $\bar{c}$, pour lequel la longueur du plus court chemin est supérieure ou égale à la longueur maximale de tout chemin efficace. Enfin, si un plus court chemin a déjà été déterminé pour les valeurs d'arcs de $\underline{c}$ à une plus grande valeur que $\bar{c}$ et la plus grande longueur d'arc correspondant est plus petite ou égale à $\bar{c}$.

Pour chaque paire $(\bar{c},\ \underline{c})$ explicitement considérée, un plus court chemin est calculé à l'étape f). Seuls ceux, parmi ces chemins, qui forment un ensemble complet de chemins non équivalents sont retenus à l'étape g). Ainsi donc, un chemin est déterminé pour tout vecteur efficace des valeurs des deux critères.

Concernant la complexité, l'étape a) requiert un temps $O(m \log n)$ pour trouver un plus court chemin P et sa longueur L^1_{opt} en utilisant l'algorithme de Dijkstra. Le temps pour déterminer le chemin d'étendue minimale dans le sous-graphe des plus courts chemins v_1-v_n, en utilisant l'algorithme MRG, est en $O(m^2 \log n)$. L'étape b) requiert un temps $O(m^2 \log n)$ pour trouver tous les couples $(\bar{c},\ \underline{c})$ qui correspondent à l'étendue minimale (il pourrait y avoir $O(m^2)$ couples, mais habituellement beaucoup moins). Ensuite, les plus courts chemins dans le sous-graphe G^* sont déterminés par l'algorithme de Dijkstra. Cela requiert en tout un temps $O(m^3 \log n)$.

Le classement des arcs à l'étape c) prend un temps $O(m \log m)$ et un temps $O(m)$ est requis pour l'initialisation. À l'étape d), un temps $O(m \log n)$ est nécessaire pour déterminer la valeur de $\underline{c}$ avec une variante de l'algorithme de Dijkstra et un temps $O(m)$ est requis pour parcourir la liste des c_i. L'étape est répétée au plus $O(m)$ fois et prend donc un temps $O(m^2 \log n)$ en tout. Les tests de l'étape e), incluant, $\bar{c} - \underline{c} \geq R^1_{opt}$, sont effectués en temps constant, et un temps $O(m)$ est requis pour parcourir la liste des c_i. Cette étape se répète au plus $O(m^2)$ fois, et requiert en tout un temps $O(m^3)$.

L'étape f) prend un temps $O(m \log n)$ pour trouver un plus court chemin v_1-v_n dans G^* avec l'algorithme de Dijkstra et cela peut se répéter $O(m^2)$ fois, i.e., cette étape requiert un temps $O(m^3 \log n)$. Le parcours de la liste des c_i prend un temps $O(m^2)$ pour chaque valeur de $\bar{c}$, et donc $O(m^3)$ en tout. La sauvegarde des chemins requiert un temps $O(m^3)$ en tout (i.e., $O(m)$ par chemin). Finalement, l'étape g) requiert un temps $O(m^2 \log m)$ pour classer les chemins et détruire ceux qui sont dominés. Les étapes dominantes sont donc b) et f) et la complexité de l'algorithme est $O(m^3 \log n)$. □

Il est clair qu'un algorithme plus simple, dans lequel les sous-graphes G^* sont construits pour tous les couples $(\bar{c}, \underline{c})$ et les plus courts chemins sont déterminés afin d'en retenir ceux qui sont non dominés, a la même complexité de pire cas que l'algorithme MRGMS. En outre, à la lumière du théorème 2.3 ci-après, il ne semble pas très facile de faire mieux en termes de complexité.

On remarquera cependant que les tests d'accélération de l'algorithme MRGMS permettent d'avoir une procédure qui devrait être significativement plus efficace en pratique que l'algorithme brut qui vient dêtre décrit. L'algorithme MRGMS est illustré sur le graphe de la figure 2.2 et les détails de la résolution sont donnés aux tableaux 2.4 et 2.5.

Théorème 2.3 *Le problème de chemin bicritère MINRANGE-MINSUM a $O(m^2)$ chemins efficaces non équivalents et cette borne est atteinte.*

Preuve. Il y a $O(m^2)$ paires de longueurs d'arcs dans G. Pour chacune de ces paires, considérée comme borne sur la longueur d'arc maximale et la longueur d'arc minimale pour un chemin v_1-v_n, il n'y a qu'une seule valeur pour la longueur du plus court chemin v_1-v_n. Ainsi, le nombre de chemins efficaces non équivalents est $O(m^2)$.

Pour montrer que cet ordre de grandeur est correct, posons, pour tout $m \geq 4$, $p = \lfloor \frac{m}{4} \rfloor$ et $n = m - 2p + 3$, et considérons le graphe G de la figure 2.3. Il possède trois sommets particuliers, v_1, v_{p+2}, v_n.

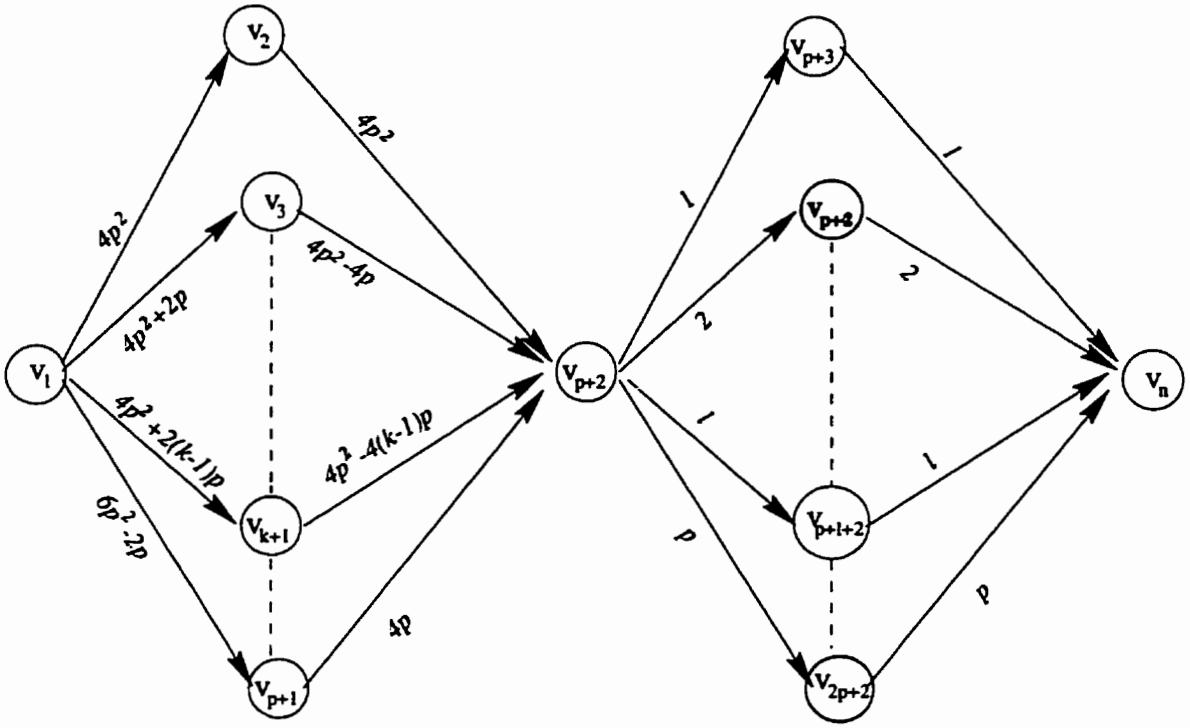


Figure 2.3 – *Graphe G pour le théorème 2.3*

Des arcs, (v_1, v_{k+1}), joignent v_1 aux p sommets v_{k+1} et ont pour longueurs $4p^2 + 2(k-1)p$ pour $k = 1, 2, ..., p$. D'autres arcs, (v_{k+1}, v_{p+2}), joignent les sommets v_{k+1} au sommet v_{p+2}. Leurs longueurs sont $4p^2 - 4(k-1)p$ avec $k = 1, 2, ..., p$. Des arcs, (v_{p+2}, v_{p+l+2}) et (v_{p+l+2}, v_n), joignent également v_{p+2} aux p sommets v_{p+l+2} et ces sommets à v_n, avec comme longueurs l, pour $l = 1, 2, ..., p$. Le graphe G est complèté par $m - 4p$ arcs allant de v_{2p+2} aux sommets pendants $v_{2p+3}, ..., v_{m-2p+2}$.

Montrons que tous les chemins $v_1 - v_n$ sont efficaces. Considérons pour cela, $P = (v_1, v_{k+1}, v_{p+2}, v_{p+l+2}, v_n)$ et $P' = (v_1, v_{k'+1}, v_{p+2}, v_{p+l'+2}, v_n)$ deux chemins distincts avec des longueurs $L(P) = 8p^2 - 2(k-1)p + 2l$ et $L(P') = 8p^2 - 2(k'-1)p + 2l'$ et des étendues $R(P) = 4p^2 + 2(k-1)p - l$ et $R(P') = 4p^2 + 2(k'-1)p - l'$ respectivement.

Si $k = k'$ alors $l \neq l'$ et, sans perte de généralité, on supposera que $l < l'$. Ainsi $L(P) - L(P') = 2l - 2l' < 0$ et $R(P) - R(P') = -l + l' > 0$. Si $k \neq k'$, on supposera, de nouveau sans perte de généralité, que $k < k'$. Alors, on obtient

$L(P) - L(P') = -2kp + 2k'p + 2l - 2l' > 0$ puisque $-p < l - l' < p$, de même que $R(P) - R(P') = 2kp - 2k'p - l + l' < 0$.

Ainsi donc, aucun chemin P de v_1 à v_n ne peut dominer un autre P'. Puisqu'il y a $p^2 = \lfloor \frac{m}{4} \rfloor^2 = O(m^2)$ chemins v_1-v_n, le résultat suit. □

On peut obtenir un algorithme pour le problème MINRATIO-MINSUM avec un schéma similaire à celui de l'algorithme MRGMS. Remarquons, pour terminer, que ces algorithmes peuvent être également utilisés lorsque l'on cherche à maximiser la fiabilité du chemin au lieu de minimiser sa longueur. En effet, le critère de la fiabilité maximale se ramène à celui de la longueur minimale en prenant comme longueur d'arc l'opposé du logarithme de la fiabilité de l'arc. On obtient cependant un problème bicritère plus difficile et plus classique lorsque l'on considère simultanément le critère de la longueur minimale et celui de la fiabilité maximale. Le chapitre suivant propose un nouvel algorithme pour ce problème.

Tableau 2.4 – *Illustration de l'algorithme MRGMS*

Itér.	a)	b)	c)	d)	e)	f)
1	$L^1_{opt}=9$ $R^1_{opt}=5$ $P^1_{opt}=(v_1,v_2,v_5)$ $\mathcal{L}=\{P^1_{opt}\}$	$R^2_{opt}=2$ $L^2_{opt}=16$ $P^2_{opt}=(v_1,v_3,v_5)$ $\mathcal{L}=\mathcal{L}\cup\{P^2_{opt}\}$	$\ell(c_i)=+\infty$, $r(c_i)=9$, $L_{opt}=+\infty$, $\overline{c}=9$, $\underline{c}^*=10$, $\underline{c}^{**}=10$ $k=1$	$\underline{c}=7$ $j=3$	$\overline{c}-\underline{c}<R^1_{opt}$ $\underline{c}<\underline{c}^*$ $\underline{c}<\underline{c}^{**}$	$P^*=P^2_{opt}$ $\ell(7)=16$ $r(7)=9$ $\underline{c}^*=7$, $\underline{c}^{**}=7$ $L_{opt}=16$ $j=5$, $\underline{c}=6$

	d)	e)	f)
2		$\overline{c}-\underline{c}<R^1_{opt}$ $\underline{c}<\underline{c}^*$, $\underline{c}<\underline{c}^{**}$	$P^*=P^2_{opt}$, $\ell(6)$, $r(6)=9$, $\underline{c}^*=6$ $\underline{c}^{**}=6$, $L_{opt}=16$, $j(6)=6$, $\underline{c}=5$
3		$\overline{c}-\underline{c}<R^1_{opt}$ $\underline{c}<\underline{c}^*$, $\underline{c}<\underline{c}^{**}$	$P^*=P^2_{opt}$, $\ell(5)=16$, $r(5)=9$, $\underline{c}^*=5$, $\underline{c}^{**}=5$, $L_{opt}=16$, $j=7$, $\underline{c}=4$
4		$\overline{c}-\underline{c}=R^1_{opt}$, $k=2$, $\overline{c}=8$, $L_{opt}=+\infty$	
5	$\underline{c}=5$ $j=6$	$\overline{c}-\underline{c}<R^1_{opt}$, $\underline{c}=\underline{c}^*$, $L_{opt}=16$, $j=7$, $\underline{c}=4$, $\overline{c}-\underline{c}<R^1_{opt}$, $\underline{c}<\underline{c}^*$, $\underline{c}<\underline{c}^{**}$	$P^*=(v_1,v_4,v_5)$, $L(P^*)=12$, $R(P^*)=4$, $\ell(4)=12$, $r(4)=8$, $\mathcal{L}=\mathcal{L}\cup\{P^*\}$, $\underline{c}^{**}=4$, $L_{opt}=12$, $j=9$, $\underline{c}=2$

Tableau 2.5 – *Illustration de l'algorithme MRGMS (suite)*

Itér	d)	e)	f)
6		$\overline{c} - \underline{c} > R^1_{opt}$, $k=3$, $\overline{c}=7$, $L_{opt}=+\infty$	
7	$\underline{c}=5$ $j=6$	$\overline{c} - \underline{c} < R^1_{opt}$, $\underline{c}=\underline{c}^*$, $L_{opt}=16$ $j=7$, $\underline{c}=4$, $\overline{c} - \underline{c} < R^1_{opt}$ $\underline{c} < \underline{c}^*$, $\underline{c}=\underline{c}^{**}$, $r(\underline{c}) > \overline{c}$	$P^*=(v_1, v_4, v_3, v_5)$, $L(P^*)=15$, $R(P^*)=3$, $\ell(4)=15$, $r(4)=7$, $d^{**}=4$ $\mathcal{L}=\mathcal{L} \cup \{P^*\}$, $L_{opt}=15$, $j=9$, $\underline{c}=2$
8		$\overline{c} - \underline{c}=R^1_{opt}$, $k=5$, $\overline{c}=6$, $L_{opt}=+\infty$	
9	$\underline{c}=2$ $j=9$	$\overline{c} - \underline{c} < R^1_{opt}$, $\underline{c} < \underline{c}^*$, $\underline{c} < \underline{c}^{**}$	$P^*=(v_1, v_4, v_3, v_2, v_5)$, $L(P^*)=16$, $R(P^*)=4$, $\ell(2)=16$, $r(2)=6$, $\underline{c}^*=2$, $\underline{c}^{**}=2$, $L_{opt}=16$, $j=10$, $\underline{c}=1$
10		$\overline{c} - \underline{c}=R^1_{opt}$, $k=6$, $\overline{c}=5$, $L_{opt}=+\infty$	

	d)	g)
11	$\mathcal{P}(\overline{G})=\emptyset$	$\mathcal{L}=\left\{(v_1, v_3, v_5); (v_1, v_4, v_3, v_5); (v_1, v_4, v_5); (v_1, v_2, v_5)\right\}$ Fin

CHAPITRE 3

UN ALGORITHME DE PLUS COURT CHEMIN BICRITÈRE

3.1 Principe et terminologie

Ce chapitre est consacré à la présentation d'un algorithme qui résout, par les deux extrémités du réseau, le problème du plus court chemin bicritère décrit à la section 1.2. L'algorithme maintient, pour chaque sommet $v_i \in V$, deux ensembles, $\mathcal{T}_{1i}$ et $\mathcal{T}_{in}$ correspondant respectivement aux étiquettes des chemins v_1-v_i et v_i-v_n temporairement efficaces. Les étiquettes efficaces permanentes des chemins v_1-v_i et v_i-v_n sont gardées dans les ensembles, $\mathcal{P}_{1i}$ et $\mathcal{P}_{in}$ respectivement, où elles sont classées lexicographiquement suivant les critère x et y, dans cet ordre.

Notons que les étiquettes de $\mathcal{P}_{in}$ et $\mathcal{P}_{1i}$ constituent des extensions potentielles pour de futures étiquettes de chemins v_1-v_i et v_i-v_n respectivement. A une itération donnée, la k-ème étiquette, par ordre lexicographique, dans l'ensemble des étiquettes efficaces permanentes des chemins v_i-v_j sera notée (X_{ij}^k, Y_{ij}^k) (voir la figure 3.1).

L'algorithme sélectionne l'étiquette de plus petite valeur lexicographique dans $\bigcup_{v_i \in V} \mathcal{T}_{1i}$ (respectivement dans $\bigcup_{v_i \in V} \mathcal{T}_{in}$) en vue d'une extension éventuelle. Cela se fait de manière que le nombre total d'étiquettes dans $\bigcup_{v_i \in V} \mathcal{T}_{1i}$ reste aussi proche que possible de celui des étiquettes dans $\bigcup_{v_i \in V} \mathcal{T}_{in}$.

Supposons que $v_{\bar{\imath}}$ est le sommet terminal du chemin correspondant à l'étiquette courante $(X_{1\bar{\imath}}, Y_{1\bar{\imath}})$ sélectionnée dans $\bigcup_{v_i \in V} \mathcal{T}_{1i}$ et soit $(X_{\bar{\jmath}n}, Y_{\bar{\jmath}n})$ la dernière étiquette

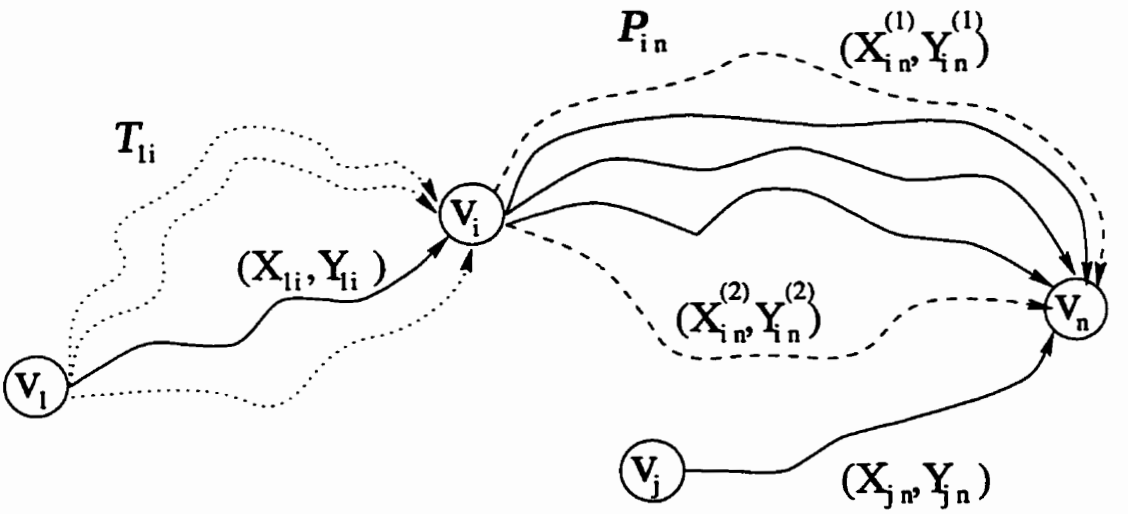


Figure 3.1 – *Illustration de l'algorithme bicritère*

sélectionnée dans $\bigcup_{v_i \in V} \mathcal{T}_{in}$. L'étiquette courante $(X_{1\bar{1}}, Y_{1\bar{1}})$ devient une étiquette efficace permanente et est déplacée de $\bigcup_{v_i \in V} \mathcal{T}_{1i}$ dans $\mathcal{P}_{1\bar{1}}$.

Les extensions de $(X_{1\bar{1}}, Y_{1\bar{1}})$ par les étiquettes déjà présentes dans $\mathcal{P}_{\bar{1}n}$ sont alors examinées en vue d'une insertion éventuelle dans l'ensemble $\mathcal{T}_{1n}$ des étiquettes de chemins v_1-v_n temporairement efficaces. Si $X_{\bar{j}n} < X_{\bar{1}n}^{(2)}$, où $X_{\bar{1}n}^{(2)}$ est la plus petite valeur de x que puisse avoir un plus court chemin $v_{\bar{1}}$-v_n pour le critère y, une approximation extérieure de l'ensemble des étiquettes efficaces permanentes des chemins $v_{\bar{1}}$-v_n est alors construite. L'extension de l'étiquette sélectionnée $(X_{1\bar{1}}, Y_{1\bar{1}})$ aux successeurs de $v_{\bar{1}}$ n'est considérée que si au moins une des étiquettes délimitant l'approximation extérieure donne une étiquette temporairement efficace au puits.

La validité de ces tests est prouvée aux propositions 3.1 et 3.2 plus bas. Si l'étiquette sélectionnée $(X_{1\bar{1}}, Y_{1\bar{1}})$ peut être prolongée aux successeurs de $v_{\bar{1}}$, alors de nouvelles étiquettes temporairement efficaces sont calculées aux différents successeurs. Le symétrique du processus est aussi effectué pour l'étiquette sélectionnée de $\bigcup_{v_i \in V} \mathcal{T}_{in}$. L'algorithme s'arrête si l'un des ensembles $\bigcup_{v_i \in V} \mathcal{T}_{1i}$ ou $\bigcup_{v_i \in V} \mathcal{T}_{in}$ est vide.

3.2 Tests de dominance

Nous décrivons d'abord un test pour la dominance d'une nouvelle étiquette. Soit un ensemble $\mathcal{L}$ contenant uniquement des étiquettes efficaces non équivalentes et triées par ordre croissant des valeurs de x (i.e., par ordre décroissant des valeurs de y). Il est clair qu'une nouvelle étiquette (X, Y), candidate à l'insertion dans $\mathcal{L}$, est dominée par rapport à $\mathcal{L}$ ou est équivalente à une étiquette déjà présente dans $\mathcal{L}$, si et seulement si $Y^* \leq Y$, où (X^*, Y^*) est l'étiquette de $\mathcal{L}$ ayant la plus grande valeur de x inférieure ou égale à X.

Si les étiquettes de $\mathcal{L}$ sont générées par ordre lexicographique, alors (X^*, Y^*) correspond à la dernière étiquette introduite et aucune suppression d'étiquette n'est requise avant l'insertion éventuelle de (X, Y). Sinon, supposons que $\mathcal{L}$ contient des étiquettes temporairement efficaces et que l'on désire utiliser (X, Y) pour faire une mise à jour de $\mathcal{L}$. Soit (X', Y') l'étiquette de $\mathcal{L}$ ayant la plus grande valeur de y inférieure ou égale à Y. Les étiquettes (X^*, Y^*) et (X', Y') délimitent le domaine des étiquettes (dominées) devant être effacées avant l'insertion de la nouvelle étiquette (X, Y).

On peut noter que si les étiquettes de $\mathcal{L}$ sont générées par ordre lexicographique, alors le test de dominance pour la nouvelle étiquette (X, Y) et son insertion dans $\mathcal{L}$ peuvent être effectués en temps constant. Sinon, en utilisant un arbre balancé, $O(\log|\mathcal{L}|))$ opérations sont requises pour trouver les étiquettes (X^*, Y^*) et (X', Y'), par une recherche dichotomique, tandis que le temps nécessaire pour supprimer les étiquettes dominées est de $O(D \log|\mathcal{L}|)$, où D est le nombre d'étiquettes à supprimer $(D < |\mathcal{L}|)$.

Nous présentons maintenant deux tests additionnels pour éliminer les étiquettes non prometteuses, i.e., dont les chemins associés ne peuvent donner des chemins v_1-v_n efficaces. La proposition 3.1 traite d'un test, qui sera dit de la sentinelle, pour déterminer si une étiquette sélectionnée peut ne pas être prolongée aux successeurs du sommet correspondant.

Proposition 3.1 *Soient le sommet $v_{\bar{\imath}}$ correspondant à l'étiquette de plus petite valeur lexicographique dans $\bigcup_{v_i \in V} \mathcal{T}_{1i}$ et $(X_{\bar{\jmath}n}, Y_{\bar{\jmath}n})$ la plus récente étiquette extraite de $\bigcup_{v_i \in V} \mathcal{T}_{in}$ pour extension. Si $X_{\bar{\jmath}n} \geq X_{\bar{\imath}n}^{(2)}$, où $X_{\bar{\imath}n}^{(2)}$ est la plus petite valeur de x d'un plus court chemin $v_{\bar{\imath}}$-v_n pour le critère y, alors $(X_{1\bar{\imath}}, Y_{1\bar{\imath}})$ peut être éliminée après avoir examiné ses extensions par les étiquettes présentes dans $\mathcal{P}_{\bar{\imath}n}$.*

Preuve. Supposons qu'il existe une étiquette $(X_{\bar{\imath}n}, Y_{\bar{\imath}n})$, associée à un chemin $v_{\bar{\imath}}$ - v_n, telle que $(X_{1\bar{\imath}}, Y_{1\bar{\imath}}) + (X_{\bar{\imath}n}, Y_{\bar{\imath}n})$ n'a pas encore été examinée. Puisque les étiquettes sélectionnées pour extension le sont par ordre lexicographique, alors $X_{\bar{\imath}n} \geq X_{\bar{\jmath}n} \geq X_{\bar{\imath}n}^{(2)}$. En remarquant que $(X_{\bar{\imath}n}^{(2)}, Y_{\bar{\imath}n}^{(2)})$ est la plus petite étiquette, par ordre lexicographique, correspondant à un plus court chemin pour le critère y, il s'ensuit que $Y_{\bar{\imath}n} \geq Y_{\bar{\imath}n}^{(2)}$. Ainsi $(X_{1\bar{\imath}}, Y_{1\bar{\imath}}) + (X_{\bar{\imath}n}^{(2)}, Y_{\bar{\imath}n}^{(2)})$ domine ou est équivalent à $(X_{1\bar{\imath}}, Y_{1\bar{\imath}}) + (X_{\bar{\imath}n}, Y_{\bar{\imath}n})$.□

Dans la proposition 3.2, ci-après, les ensembles $\mathcal{P}_{in}$ sont supposés contenir uniquement les étiquettes efficaces permanentes. En outre, on considère qu'ils sont initialisés avec, au moins, les étiquettes de plus petite valeur lexicographique correspondant aux plus court chemins v_i-v_n pour chacun des critère, i.e., $(X_{in}^{(1)}, Y_{in}^{(1)})$ et $(X_{in}^{(2)}, Y_{in}^{(2)})$. Cette proposition définit un autre test qui permet d'éliminer une étiquette non prometteuse, même si elle est localement efficace.

Proposition 3.2 *Soient $(X_{1\bar{\imath}}, Y_{1\bar{\imath}})$, l'étiquette de plus petite valeur lexicographique dans $\bigcup_{v_i \in V} \mathcal{T}_{1i}$, et $(X_{\bar{\imath}n}^{k}, Y_{\bar{\imath}n}^{k})$, pour $k = 1, 2, ..., |\mathcal{P}_{\bar{\imath}n}|$, la k-ème étiquette connue de l'ensemble $\mathcal{P}_{\bar{\imath}n}$ correspondant. Si toutes les étiquettes $(X_{1\bar{\imath}}, Y_{1\bar{\imath}}) + (X_{\bar{\imath}n}^{k}, Y_{\bar{\imath}n}^{k+1})$, pour $k = 1, 2, ..., |\mathcal{P}_{\bar{\imath}n}| - 1$, sont dominées par rapport à $\mathcal{T}_{1n} \cup \mathcal{P}_{1n}$, alors alors il n'est pas nécessaire de prolonger $(X_{1\bar{\imath}}, Y_{1\bar{\imath}})$.*

Preuve. Supposons, par l'absurde, qu'il existe une étiquette $(X_{\bar{\imath}n}, Y_{\bar{\imath}n})$, correspondant à un chemin $v_{\bar{\imath}}$ - v_n, telle que $(X_{1\bar{\imath}}, Y_{1\bar{\imath}}) + (X_{\bar{\imath}n}, Y_{\bar{\imath}n})$ est efficace par rapport à $\mathcal{T}_{1n} \cup \mathcal{P}_{1n}$ mais n'a pas encore été examinée. Puisqu'un chemin v_1-v_n efficace contient uniquement des sous-chemins efficaces, alors $(X_{\bar{\imath}n}, Y_{\bar{\imath}n})$ est nécessairement efficace par rapport à $\mathcal{P}_{\bar{\imath}n}$.

Soit alors $(X_{\bar{\imath}n}^{k^*}, Y_{\bar{\imath}n}^{k^*})$ l'étiquette courante de $\mathcal{P}_{\bar{\imath}n}$ ayant la plus grande valeur

de x inférieure ou égale à $X_{\bar{\imath}n}$. On a $X_{\bar{\imath}n}^{k^*} \leq X_{\bar{\imath}n} \leq X_{\bar{\imath}n}^{k^*+1}$ et, puisque $\mathcal{P}_{\bar{\imath}n}$ contient uniquement des étiquettes efficaces, $Y_{\bar{\imath}n}^{k^*+1} \leq Y_{\bar{\imath}n} \leq Y_{\bar{\imath}n}^{k^*}$. Donc, en supposant que $(X_{1\bar{\imath}}, Y_{1\bar{\imath}}) + (X_{\bar{\imath}n}^{k^*}, Y_{\bar{\imath}n}^{k^*+1})$ est dominé par rapport à $\mathcal{T}_{1n} \cup \mathcal{P}_{1n}$ il s'en suit que $(X_{1\bar{\imath}}, Y_{1\bar{\imath}}) + (X_{\bar{\imath}n}, Y_{\bar{\imath}n})$ est aussi dominé par rapport à $\mathcal{T}_{1n} \cup \mathcal{P}_{1n}$, d'où une contradiction. □

On peut remarquer que les étiquettes $(X_{\bar{\imath}n}^k, Y_{\bar{\imath}n}^{k+1})$, pour $k = 1, 2, ..., |\mathcal{P}_{\bar{\imath}n}| - 1$, définissent, avec les étiquettes de plus courts chemins lexicographiques $(X_{\bar{\imath}n}^{(1)}, Y_{\bar{\imath}n}^{(1)})$ et $(X_{\bar{\imath}n}^{(2)}, Y_{\bar{\imath}n}^{(2)})$, une approximation extérieure de l'ensemble des étiquettes efficaces (permanentes) correspondant à des chemins $v_{\bar{\imath}}$-v_n. En effet, étant donnée une telle étiquette efficace, $(X_{\bar{\imath}n}, Y_{\bar{\imath}n})$, distincte de $(X_{\bar{\imath}n}^{(1)}, Y_{\bar{\imath}n}^{(1)})$ et de $(X_{\bar{\imath}n}^{(2)}, Y_{\bar{\imath}n}^{(2)})$ qui sont supposées avoir été utilisées pour initialiser $\mathcal{P}_{\bar{\imath}n}$, les arguments dans la preuve ci-dessus impliquent qu'il existe une étiquette $(X_{\bar{\imath}n}^{k^*}, Y_{\bar{\imath}n}^{k^*}) \in \mathcal{P}_{\bar{\imath}n}$ telle que $X_{\bar{\imath}n}^{k^*} \leq X_{\bar{\imath}n}$ et $Y_{\bar{\imath}n}^{k^*+1} \leq Y_{\bar{\imath}n}$.

Ainsi, la proposition 3.2 revient à tester la dominance, par rapport à $\mathcal{T}_{1n} \cup \mathcal{P}_{1n}$, pour les points extrêmes d'une approximation extérieure de $\mathcal{P}_{\bar{\imath}n}$ translatée par l'étiquette courante sélectionnée $(X_{1\bar{\imath}}, Y_{1\bar{\imath}})$ (voir la figure 3.2). Il est clair que plus $|\mathcal{P}_{\bar{\imath}n}|$ augmente, meilleure sera l'approximation.

Notons également qu'étant donnée l'étiquette courante sélectionnée $(X_{1\bar{\imath}}, Y_{1\bar{\imath}})$ et le sommet correspondant $v_{\bar{\imath}}$, les tests proposés par Tung et Chew, dans [57] et dans [58], pour éliminer $(X_{1\bar{\imath}}, Y_{1\bar{\imath}})$ reviennent à déterminer des approximations extérieures de $\mathcal{P}_{jn}$, aux successeurs v_j de $v_{\bar{\imath}}$, en utilisant au plus une étiquette autre que $(X_{jn}^{(1)}, Y_{jn}^{(1)})$ et $(X_{jn}^{(2)}, Y_{jn}^{(2)})$. Il s'agit en l'occurence de l'étiquette fictive dont les valeurs de x et y correspondent aux plus courts chemins v_j-v_n pour les critères x et y respectivement, ou de l'étiquette efficace extrême de $\mathcal{P}_{jn}$ correspondant à la somme des deux critères.

La fonction d'évaluation utilisée par ces auteurs pour sélectionner l'étiquette à traiter, $(X_{1\bar{\imath}}, Y_{1\bar{\imath}})$, est la somme des deux fonctions objectifs. Cependant, l'étiquette sélectionnée $(X_{1\bar{\imath}}, Y_{1\bar{\imath}})$ peut être prolongée en v_j, même si elle est dominée par une étiquette déjà trouvée en $v_{\bar{\imath}}$. De ce fait, l'algorithme proposé par ces auteurs n'utilise

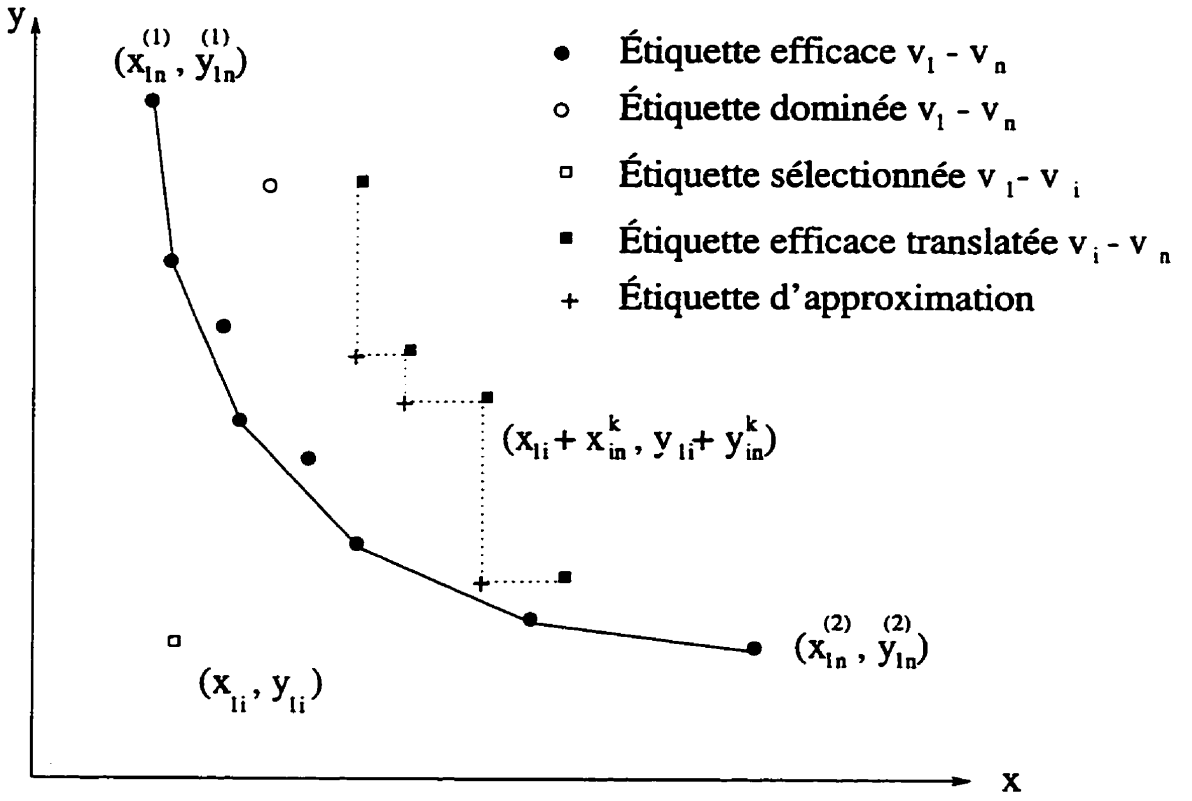


Figure 3.2 – *Approximation extérieure pour les étiquettes v_i-v_n au sommet v_i*

pas la propriété spécifiant qu'un chemin v_1-v_n efficace ne contient que des sous-chemins efficaces (voir Hansen [36] ou Martins [47]).

De toute évidence, l'analogue de chacune des propositions 3.1 et 3.2 est vraie pour l'étiquette, $(X_{\bar{j}n}, Y_{\bar{j}n})$, de plus petite valeur lexicographique dans $\bigcup_{v_i \in V} \mathcal{T}_{in}$.

3.3 Initialisation

Soit un sommet v_i, le schéma d'initialisation de base de l'algorithme consiste à déterminer les étiquettes de plus petite valeur lexicographique, $(X^{(1)}_{1i}, Y^{(1)}_{1i})$, $(X^{(2)}_{1i}, Y^{(2)}_{1i})$, $(X^{(1)}_{in}, Y^{(1)}_{in})$ et $(X^{(2)}_{in}, Y^{(2)}_{in})$ correspondant aux plus courts chemins v_1-v_i et v_i-v_n, pour chaque critère. Un plus court chemin v_1-v_i par ordre lexicographique,

peut être trouvé en utilisant une variante de l'algorithme de Dijkstra [25].

Il suffit, pour cela, d'effectuer la comparaison des étiquettes en utilisant l'ordre lexicographique sur (X, Y) (ou sur (Y, X), pour le critère y). Les plus courts chemins v_i-v_n peuvent être déterminés d'une façon similaire en parcourant le graphe dans le sens contraire des arcs.

Une initialisation plus étendue peut être effectuée en déterminant d'autres étiquettes efficaces extrêmes correspondant à des chemins v_1-v_n. De telles étiquettes peuvent être trouvées en utilisant une méthode de plus courts chemins paramétriques, basée sur des combinaisons convexes des deux critères: $\alpha x_{ij} + (1-\alpha) y_{ij}$, où $\alpha \in [0, 1]$.

On peut considérer seulement quelques valeurs de α prédéfinies, ou toutes les valeurs nécessaires pour déterminer l'ensemble des étiquettes efficaces extrêmes. Une façon de procéder consiste à examiner une suite de subdivisions binaires de l'ensemble des étiquettes efficaces extrêmes.

En effet, soient (X', Y') et (X'', Y'') deux étiquettes efficaces extrêmes distinctes. Considérons le problème de l'énumération de toutes les étiquettes efficaces extrêmes ayant des valeurs de x comprises entre X' et X''. Les deux étiquettes induisent une combinaison convexe des deux critères, avec pour paramètres $\alpha = (Y' - Y'')/(X'' - X' + Y' - Y'')$.

Soit (X, Y) l'étiquette optimale du problème de plus court chemin correspondant. Selon le théorème 3.3 ci-dessous, (X, Y) est une étiquette efficace extrême et deux nouveaux sous-problèmes d'énumération, définis par les étiquettes efficaces extrêmes (X', Y') et (X, Y) pour la première et par (X, Y) et (X'', Y'') pour la seconde, doivent être considérés.

Il n'est pas nécessaire d'explorer un sous-problème donné si les deux étiquettes qui le définissent ont la même valeur pour l'objectif combiné ayant (X, Y) comme étiquette optimale. Cette exploration dichotomique de l'ensemble des étiquettes efficaces extrêmes peut être effectuée en utilisant, par exemple, une technique de recherche en profondeur d'abord.

Théorème 3.3 *Soient (X', Y') et (X'', Y'') deux étiquettes efficaces extrêmes telles que $X' < X''$ et soit (X, Y) l'étiquette optimale du problème de plus court chemin paramétrique pour $\alpha = (Y' - Y'')/(X'' - X' + Y' - Y'')$. Alors (X, Y) est une étiquette efficace extrême et $X' \leq X \leq X''$. En outre, si $\alpha X + (1-\alpha)Y = \alpha X' + (1-\alpha)Y'$, aucune autre étiquette efficace extrême n'est située entre (X', Y') et (X, Y), ni entre (X, Y) et (X'', Y'').*

Preuve. Notons que l'hypothèse $X' < X''$ n'implique aucune perte de généralité, mais entraîne que $\alpha \in]0, 1[$, puisque (X', Y') et (X'', Y'') sont des étiquettes efficaces et ainsi $Y' > Y''$. Par conséquent, le problème de plus court chemin paramétrique pour α est bien défini avec, en particulier, $\alpha X + (1-\alpha)Y \leq \alpha X'' + (1-\alpha)Y''$, $\alpha X' + (1-\alpha)Y' = \alpha X'' + (1-\alpha)Y''$ et (X, Y) étant une étiquette efficace extrême (voir, par exemple, White [63]).

Nous montrons maintenant que $X' \leq X \leq X''$. En effet, en supposant $X < X'$, on a $Y > Y'$ par non dominance et il existe $\lambda \in]0, 1[$ avec $X' = \lambda X + (1-\lambda)X''$ et $Y' \leq \lambda Y + (1-\lambda)Y''$, par convexité de l'ensemble des étiquettes efficaces extrêmes. De ce fait, $\alpha X' + (1-\alpha)Y' \leq \alpha\lambda X + \alpha(1-\lambda)X'' + (1-\alpha)(\lambda Y + (1-\lambda)Y'') = \lambda(\alpha X + (1-\alpha)Y) + (1-\lambda)(\alpha X'' + (1-\alpha)Y'') = \lambda(\alpha X + (1-\alpha)Y) + (1-\lambda)(\alpha X' + (1-\alpha)Y')$, i.e. $\lambda(\alpha X' + (1-\alpha)Y') \leq \lambda(\alpha X + (1-\alpha)Y)$.

Ainsi, $\alpha X' + (1-\alpha)Y' < \alpha X + (1-\alpha)Y$, puisque l'égalité impliquerait que (X, Y), (X', Y') et (X'', Y'') sont sur la même droite, ce qui contredirait le fait que les trois étiquettes sont toutes des points extrêmes de l'enveloppe convexe des étiquettes efficaces, dans l'espace des étiquettes. On obtient donc une contradiction de l'optimalité de (X, Y) pour le problème de plus court chemin paramétrique associé à α et donc $X' \leq X$. De manière similaire, on peut montrer que $X \leq X''$.

Nous discutons maintenant l'existence d'autres étiquettes efficaces entre (X', Y') et (X'', Y''). Aucune autre étiquette efficace n'est située entre (X', Y') et (X, Y) si ces deux étiquettes sont égales. Supposons donc qu'elles sont distinctes et qu'il existe, entre elles, une étiquette efficace extrême (X^*, Y^*) distincte de chacune d'elles. On a alors $X' < X^* < X$ et $Y < Y^* < Y'$. Par ailleurs, il existe $\lambda \in]0, 1[$ tel que $X^* = \lambda X' + (1-\lambda)X$ et $Y^* \leq \lambda Y' + (1-\lambda)Y$, par convexité de l'ensemble des

étiquettes efficaces extrêmes.

Par conséquent, si $\alpha X + (1-\alpha)Y = \alpha X' + (1-\alpha)Y'$, nous avons $\alpha X^* + (1-\alpha)Y^* \leq \alpha\lambda X' + \alpha(1-\lambda)X + (1-\alpha)(\lambda Y' + (1-\lambda)Y) = \lambda(\alpha X' + (1-\alpha)Y') + (1-\lambda)(\alpha X + (1-\alpha)Y) = \lambda(\alpha X + (1-\alpha)Y) + (1-\lambda)(\alpha X + (1-\alpha)Y) = \alpha X + (1-\alpha)Y$, i.e. $\alpha X^* + (1-\alpha)Y^* < \alpha X + (1-\alpha)Y$ puisque l'égalité impliquerait que les étiquettes efficaces extrêmes (X', Y'), (X^*, Y^*) et (X, Y) sont sur la même droite.

Ainsi, on obtient une contradiction de l'optimalité de (X, Y) pour le problème de plus court chemin paramétrique associé à α. Il n'existe donc aucune autre étiquette efficace extrême entre (X', Y') et (X, Y) si $\alpha X + (1-\alpha)Y = \alpha X' + (1-\alpha)Y'$. De manière similaire, aucune autre étiquette efficace extrême n'est située entre (X, Y) et (X'', Y'') puisque $\alpha X' + (1-\alpha)Y' = \alpha X'' + (1-\alpha)Y''$. □

Il est clair que si aucune autre étiquette efficace extrême n'est située entre (X', Y') et (X, Y), alors les deux étiquettes sont optimales pour le nouveau problème de plus court chemin qu'elles définissent. Aucune autre étiquette efficace extrême n'est optimale pour ce nouveau problème, puisque le contraire signifierait que les trois étiquettes efficaces extrêmes sont situées sur la même droite. De ce fait, au plus un problème de plus court chemin paramétrique doit être résolu, pour sonder le sous-problème défini par (X', Y') et (X, Y), dans ce cas particulier. Il en est de même pour (X, Y) et (X'', Y'').

Soit γ le nombre d'étiquettes efficaces extrêmes situées entre les deux étiquettes originales (X', Y') et (X'', Y''). Il suffit donc de résoudre $2\gamma + 1$ problèmes de plus court chemin paramétrique pour sonder le sous-problème défini par (X', Y') et (X'', Y''). Remarquons que cette procédure n'implique aucune dégénérescence potentielle, comme c'est le cas lorsqu'un argument de type simplicial est utilisé pour ajuster le paramètre α (voir, par exemple, Henig [38] ou Mote *et al.* [51]).

Notons également que certaines étiquettes efficaces extrêmes correspondant à des chemins v_1-v_i, pour $v_i \in V$, peuvent être générées, comme sous-produits de ces calculs de plus courts chemins paramétriques. On peut ainsi obtenir, éventuellement, une

meilleure initialisation, des ensembles $\mathcal{P}_{1i}$. Une telle initialisation peut être également effectuée pour les ensembles $\mathcal{P}_{in}$, en déterminant, de manière similaires, des plus courts chemins paramétriques du puits à la source, par une traversée du graphe dans le sens contraire de celui des arcs.

3.4 Énoncé de l'algorithme

Dans l'algorithme MSMS ci-après, chaque test de dominance implique aussi un test pour une copie déjà existante de l'étiquette candidate considérée. Pour des raisons de simplicité, nous ne décrivons pas les pointeurs nécessaires pour reconstruire le chemin correspondant à une étiquette efficace donnée. Un tel pointeur doit, en principe être spécifié chaque fois qu'une étiquette temporaire est calculée.

Algorithme MSMS (MINSUM-MINSUM)

1. *Initialisation*

   (a) Effectuer l'**Initialisation de base** ou l'**Initialisation étendue**.

   (b) Pour $v_i \in V - \{v_1, v_n\}$, $(X, Y) \in \mathcal{P}_{1i}$ et $(X', Y') \in \mathcal{P}_{in}$, supprimer toutes les étiquettes de $\mathcal{T}_{1n}$ qui sont dominées par $(X, Y) + (X', Y')$ et ajouter cette nouvelle étiquette à $\mathcal{T}_{1n}$ si elle est efficace par rapport à $\mathcal{T}_{1n}$ et à $\mathcal{P}_{1n}$.

2. *Étape principale*
   Tant que $\bigcup_{v_i \in V} \mathcal{T}_{1i} \neq \emptyset$ et $\bigcup_{v_i \in V} \mathcal{T}_{in} \neq \emptyset$: effectuer la **Phase en avant** si $|\bigcup_{v_i \in V} \mathcal{T}_{1i}| \leq |\bigcup_{v_i \in V} \mathcal{T}_{in}|$, sinon effectuer la **Phase en arrière**.

Initialisation de base

1. *Plus courts chemins à un critère*
   Déterminer les étiquettes de plus courts chemins lexicographiques $(X_{1i}^{(1)}, Y_{1i}^{(1)})$, $(X_{in}^{(1)}, Y_{in}^{(1)})$, $(X_{1i}^{(2)}, Y_{1i}^{(2)})$, et $(X_{in}^{(2)}, Y_{in}^{(2)})$, pour tout $v_i \in V$.

2. *Initialisation des ensembles d'étiquettes*
   Poser $X_{1\mathfrak{r}} = 0$, $\mathcal{P}_{11} = \mathcal{T}_{11} = \{(X_{11}^{(1)}, Y_{11}^{(1)})\}$ et, pour tout $v_i \in V - \{v_1\}$, $\mathcal{T}_{1i} = \emptyset$ et $\mathcal{P}_{1i} = \{(X_{1i}^{(1)}, Y_{1i}^{(1)}), X_{1i}^{(2)}, Y_{1i}^{(2)})\}$.
   Poser également $X_{\mathfrak{jn}} = 0$, $\mathcal{P}_{nn} = \mathcal{T}_{nn} = \{(X_{nn}^{(1)}, Y_{nn}^{(1)})\}$ et, pour tout $v_j \in V - \{v_1, v_n\}$, $\mathcal{T}_{jn} = \emptyset$ et $\mathcal{P}_{jn} = \{(X_{jn}^{(1)}, Y_{jn}^{(1)}), (X_{jn}^{(2)}, Y_{jn}^{(2)})\}$.

Initialisation étendue Initialisation étendue

1. Effectuer l'**Initialisation de base** et poser $\mathcal{C} = \{(X_{1n}^{(1)}, Y_{1n}^{(1)}, X_{1n}^{(2)}, Y_{1n}^{(2)})\}$.

2. Tant que $\mathcal{C} \neq \emptyset$:

   (a) Retirer un élément (X', Y', X'', Y'') de $\mathcal{C}$;
   calculer $\alpha = (Y' - Y'')/(X'' - X' + Y' - Y'')$ et ajuster les coûts des arcs à $\alpha x_{ij} + (1 - \alpha)y_{ij}$, pour tout $(v_i, v_j) \in A$.

   (b) Pour $v_i \in V$, déterminer les étiquettes des plus courts chemins v_1-v_i et v_i-v_n et les ajouter à $\mathcal{P}_{1i}$ et à $\mathcal{P}_{in}$ respectivement;

   (c) soit (X, Y) l'étiquette d'un plus court chemin v_1-v_n;
   si $\alpha X + (1-\alpha)Y < \alpha X' + (1-\alpha)Y'$, ajouter (X', Y', X, Y) et (X, Y, X'', Y'') à l'ensemble $\mathcal{C}$.

Phase en avant

1. *Sélection d'une étiquette de chemin v_1-$v_{\mathfrak{r}}$*
   Retirer de $\bigcup_{v_i \in V} \mathcal{T}_{1i}$ une étiquette, $(X_{1\mathfrak{r}}, Y_{1\mathfrak{r}})$, de plus petite valeur lexicographique et l'insérer dans $\mathcal{P}_{1\mathfrak{r}}$.

2. *Extension de l'étiquette de chemin v_1-$v_{\mathfrak{r}}$ sélectionnée*

   (a) Pour $(X, Y) \in \mathcal{P}_{\mathfrak{r}n}$, supprimer toutes les étiquettes de $\mathcal{T}_{1n}$ qui sont dominées par $(X_{1\mathfrak{r}}, Y_{1\mathfrak{r}}) + (X, Y)$ et ajouter cette nouvelle étiquette à $\mathcal{T}_{1n}$ si elle est efficace par rapport à $\mathcal{T}_{1n}$ et à $\mathcal{P}_{1n}$.

(b) Si $X_{\bar{j}n} < X^{(2)}_{\bar{i}n}$ et au moins une étiquette $(X_{1\bar{i}}, Y_{1\bar{i}}) + (X^k_{\bar{i}n}, Y^{k+1}_{\bar{i}n})$, pour $k = 1, 2, ..., |\mathcal{P}_{\bar{i}n}| - 1$, est efficace par rapport à $\mathcal{T}_{1n}$ et à $\mathcal{P}_{1n}$, alors, pour $(v_{\bar{i}}, v_j) \in A$: supprimer toutes les étiquettes de $\mathcal{T}_{1j}$ qui sont dominées par $(X_{1\bar{i}}, Y_{1\bar{i}}) + (x_{\bar{i}j}, y_{\bar{i}j})$ et ajouter cette nouvelle étiquette à $\mathcal{T}_{1j}$ si elle est efficace par rapport à $\mathcal{T}_{1j}$ et à $\mathcal{P}_{1j}$.

Phase en arrière

1. *Sélection d'une étiquette de chemin $v_{\bar{j}}$ - v_n*
   Retirer de $\bigcup_{v_i \in V} \mathcal{T}_{in}$ une étiquette, $(X_{\bar{j}n}, Y_{\bar{j}n})$, de plus petite valeur lexicographique et l'insérer dans $\mathcal{P}_{\bar{j}n}$.

2. *Extension de l'étiquette de chemin $v_{\bar{j}}$ - v_n sélectionnée*

   (a) Pour $(X, Y) \in \mathcal{P}_{1\bar{j}}$, supprimer toutes les étiquettes de $\mathcal{T}_{1n}$ qui sont dominées par $(X, Y) + (X_{\bar{j}n}, Y_{\bar{j}n})$ et ajouter cette nouvelle étiquette à $\mathcal{T}_{1n}$ si elle est efficace par rapport à $\mathcal{T}_{1n}$ et à $\mathcal{P}_{1n}$.

   (b) Si $X_{1\bar{i}} < X^{(2)}_{1\bar{j}}$ et au moins une étiquette $(X^k_{1\bar{j}}, Y^{k+1}_{1\bar{j}}) + (X_{\bar{j}n}, Y_{\bar{j}n})$, pour $k = 1, 2, ..., |\mathcal{P}_{1\bar{j}}| - 1$, est efficace par rapport à $\mathcal{T}_{1n}$ et à $\mathcal{P}_{1n}$, alors, pour $(v_i, v_{\bar{j}}) \in A$: supprimer toutes les étiquettes de $\mathcal{T}_{in}$ qui sont dominées par $(x_{i\bar{j}}, y_{i\bar{j}}) + (X_{\bar{j}n}, Y_{\bar{j}n})$ et ajouter cette nouvelle étiquette à $\mathcal{T}_{in}$ si elle est efficace par rapport à $\mathcal{T}_{in}$ et à $\mathcal{P}_{in}$.

3.5 Justification de l'algorithme

On peut remarquer qu'à une itération donnée, toute nouvelle étiquette entrant dans $\bigcup_{v_i \in V} \mathcal{T}_{1i}$ (respectivement dans $\bigcup_{v_i \in V} \mathcal{T}_{in}$) est obtenue par extension de l'étiquette de plus petite valeur lexicographique sélectionnée de $\bigcup_{v_i \in V} \mathcal{T}_{1i}$ (respectivement de $\bigcup_{v_i \in V} \mathcal{T}_{in}$). La propriété suivante est, dès lors, immédiate, puisque les valeurs des arcs sont non-négatives pour les deux critères.

Propriété 3.4 *Étant donné un sommet $v_i \in V$, soient (X_{1i}, Y_{1i}) et (X'_{1i}, Y'_{1i}), respectivement, les étiquettes de plus petites valeurs lexicographiques sélectionnées de $\bigcup_{v_i \in V} \mathcal{T}_{1i}$ aux itérations k et k' telles que $k > k'$, alors (X_{1i}, Y_{1i}) est lexicographiquement plus grande ou égale à (X'_{1i}, Y'_{1i}). L'analogue est également vraie pour $\bigcup_{v_i \in V} \mathcal{T}_{in}$.*

A un sommet donné $v_i \in V$, les ensembles $\mathcal{P}_{1i}$ et $\mathcal{P}_{in}$ sont initialisés avec des étiquettes efficaces extrêmes. La propriété ci-dessus implique que toutes les étiquettes futures, entrant dans l'un de ces ensembles, sont générées par ordre lexicographique croissant. Étant donné qu'un test de dominance est effectué par rapport à $\mathcal{P}_{1i}$ (respectivement $\mathcal{P}_{in}$) avant d'introduire toute nouvelle étiquette dans $\bigcup_{v_i \in V} \mathcal{T}_{1i}$ (respectivement dans $\bigcup_{v_j \in V} \mathcal{T}_{jn}$), la propriété ci-après est également triviale.

Propriété 3.5 *Une étiquette, de plus petite valeur lexicographique, sélectionnée de $\bigcup_{v_i \in V} \mathcal{T}_{1i}$ ou de $\bigcup_{v_j \in V} \mathcal{T}_{jn}$ est une étiquette efficace permanente au sommet correspondant.*

La propriété 3.5 implique que les ensembles $\mathcal{P}_{1i}$ et $\mathcal{P}_{in}$, pour $v_i \in V$, contient uniquement des étiquettes efficaces permanentes à toute itération de l'algorithme. Le résultat suivant justifie le test d'arrêt de l'algorithme

Proposition 3.6 *Si, au début d'une itération, $\bigcup_{v_i \in V} \mathcal{T}_{1i} = \emptyset$ ou $\bigcup_{v_i \in V} \mathcal{T}_{in} = \emptyset$, alors $\mathcal{P}_{1n}$ contient toutes, et seulement, les étiquettes efficaces correspondant aux chemins v_1-v_n.*

Preuve Considérons le cas où $\bigcup_{v_i \in V} \mathcal{T}_{1i} = \emptyset$. Commençant à v_1 avec l'étiquette $(0, 0)$, toutes les étiquettes pouvant être obtenues à partir d'une étiquette efficace sélectionnée de $\bigcup_{v_i \in V} \mathcal{T}_{1i}$ sont examinées et introduites dans cet ensemble, à moins que qu'elle soit dominée ou qu'une copie équivalente existe. Les étiquettes $\bigcup_{v_i \in V} \mathcal{T}_{1i}$ sont sélectionnée par ordre lexicographique (voir la propriété 3.4) jusqu'à ce qu'il n'en reste plus.

Ainsi, chaque étiquette efficace correspondant à un chemin v_1-v_n doit avoir été calculée puisque le chemin correspondant ne contient que des sous-chemins efficaces. L'analyse est similaire si $\bigcup_{v_i \in V} \mathcal{T}_{in}$ est vide. Puisque toute étiquette sélectionnée est efficace (voir la propriété 3.5), le résultat suit.□

Le théorème suivant établit l'exactitude de l'algorithme MSMS et donne sa complexité.

Théorème 3.7 *L'algorithme MSMS détermine l'ensemble des étiquettes efficaces correspondant à des chemins v_1-v_n. Si le critère x ne prend que des valeurs entières non-négatives, alors l'algorithme requiert un temps $O(n^4 D^3 \log nD)$, où D est la plus grande valeur de x sur un arc.*

Preuve. Étant donnée la propriété 3.5, l'exactitude de l'algorithme est immédiate par les propositions 3.1, 3.2 et 3.6. Pour la complexité, nous supposerons que l'initialisation de base est utilisée. Ainsi, l'étape 1a requiert $O(n^2)$ opérations si une variante de l'algorithme de Dijkstra est utilisée, puisque la seule modification consiste à utiliser l'ordre lexicographique lors de la comparaison des étiquettes.

A l'étape 1b, on peut remarquer qu'il y a $O(nD)$ valeurs du critère x pour un chemin élémentaire arrivant ou quittant un sommet donné, i.e, pour les étiquettes de $\mathcal{P}_{1i}$ ou de $\mathcal{P}_{in}$ en tout sommet v_i. Par conséquent, les tests de dominance et les insertions à cette étape prennent un temps $O(n^3 D^2 \log nD)$ en tout (voir la section 3.2). De même, il y a $O(n^2 D)$ étiquettes efficaces en tout, pour les chemins quittant la source ou arrivant au puits, et ainsi, l'étape 2 se répète $O(n^2 D)$ fois.

Considérons maintenant la phase "en avant" et supposons que les sommets sont contenus dans une queue de priorité selon leurs étiquettes de plus petites valeurs lexicographiques dans les ensembles $\mathcal{T}_{1i}$ correspondants. A l'étape 1, une nouvelle étiquette peut être sélectionnée de $\bigcup_{v_i \in V} \mathcal{T}_{1i}$ en $O(\log n)$ opérations et les ensembles $\mathcal{P}_{1i}$ et $\mathcal{T}_{1i}$ correspondant peuvent être mis à jour en temps $O(\log nD)$ (voir la section 3.2). La complexité de l'étape 1 est alors $O(\log nD)$.

Étant donné qu'il y a $O(nD)$ étiquettes dans $\mathcal{P}_{in}$, les tests de dominance et les insertions, à l'étape 2a, nécessitent un temps $O(n^2D^2 \log nD)$. $O(nD)$ tests de dominance, impliquant $O(\log nD)$ opérations chacune, sont nécessaires pour décider si les successeurs de v_i doivent être examinés à l'étape 2b.

Le traitement de chacun des $O(n)$ successeurs requiert $O(nD \log nD)$ opérations, puisque $O(\log nD)$ opérations sont nécessaires pour tester la dominance, alors que $\mathcal{T}_{1j}$ prend un temps $O(nD \log nD)$ pour être mis à jour. Par conséquent, la complexité de l'étape 2b est $O(n^2D \log nD)$ et celle de la phase "en avant" est $O(n^2D^2 \log nD)$, puisque la mise à jour de la queue de priorité des sommets, à la fin de cette phase, ne requiert qu'un temps $O(n \log n)$.

De manière similaire, on peut montrer que la phase "en arrière" a la même complexité. L'étape dominante est 2a dans chacune des phases "en avant" et "en arrière". Puisque cette étape requiert un temps $O(n^2D^2 \log nD)$ à chacune des $O(n^2D)$ itérations de l'étape principale de l'algorithme, le résultat suit.□

On peut noter qu'à chaque itération, l'algorithme examine, dans le pire cas, toutes les étiquettes de $\mathcal{P}_{in}$ ou de $\mathcal{P}_{1j}$. Ces étiquettes caractérisent les sous-chemins efficaces, déjà calculés au sommet correspondant à l'étiquette sélectionnée. Ceci permet de mettre à jour l'ensemble des étiquettes temporairement efficaces de chemins v_1-v_n et d'éliminer éventuellement certaines étiquettes (efficaces) sélectionnées qui ne peuvent donner d'étiquettes efficaces de chemins v_1-v_n.

La complexité d'un algorithme similaire prolongeant les étiquettes sélectionnées sans utiliser les étiquettes de sous-chemins efficaces déjà calculées pour éliminer les étiquettes non prometteuses est $O(n^4D^2 \log nD)$. Cependant, un tel algorithme est susceptible d'avoir une moins bonne performance en pratique, en particulier lorsque la densité ou la taille du réseau augmente.

3.6 Extension

L'algorithme proposé peut aisément être modifié pour générer les étiquettes situées dans un rectangle donné (i.e. défini par des bornes inférieures et supérieures sur chacun des critères). Une première modification consiste à vérifier, pendant le test de dominance par rapport à un ensemble donné d'étiquettes efficaces permanentes ou temporaires:(i) si la nouvelle étiquette est dans le rectangle spécifié lorsqu'il s'agit d'un chemin v_1-v_n ou,(ii) pour tout autre chemin, si les valeurs de x et de y sont inférieures ou égales aux bornes supérieures spécifiées sur les critères.

Une autre modification consiste à initialiser $\mathcal{P}_{1n}$ avec deux étiquettes efficaces extrêmes qui ne sont pas dans le rectangle mais sont, de préférence, le plus près possible des limites de celui-ci. Ces deux étiquettes doivent, bien entendu, être enlevées de l'ensemble $\mathcal{P}_{1n}$ à la fin de l'algorithme. Notons que certaines étiquettes efficaces non extrêmes pourraient ne pas être trouvées si l'algorithme est initialisé uniquement avec des étiquettes efficaces extrêmes appartenant au rectangle. Ce ne sera cependant pas le cas si les bornes spécifiées englobent l'étiquette de plus petite valeur lexicographique pour chaque critère.

Les deux étiquettes nécessaires pour initialiser l'ensemble $\mathcal{P}_{1n}$ peuvent être déterminées en utilisant le schéma d'énumération du théorème 3.3. Il suffit d'ajouter un test supplémentaire pour vérifier si au moins une des deux étiquettes définissant tout sous-problème d'énumération appartient au rectangle, avant de résoudre le problème de plus court chemin paramétrique. Un sous-problème défini par des étiquettes situées toutes les deux dans le rectangle peut, cependant, être sauté, puisqu'il conduit nécessairement à des étiquettes efficaces extrêmes contenues dans le rectangle.

On résoudra donc, de préférence, uniquement les sous-problèmes dont seulement une des étiquettes de définition est dans le rectangle. La procédure peut générer éventuellement d'autres étiquettes efficaces extrêmes de chemins v_1-v_n, qui soient admissibles. Elles peuvent, par conséquent, être ajoutées à $\mathcal{P}_{1n}$ avant d'exécuter la phase principale de l'algorithme. Notons également qu'un schéma d'énumération qui génère les étiquettes efficaces extrêmes par ordre croissant de l'un des critères (par

exemple, dans Henig [38]) serait moins performante pour trouver les étiquettes efficaces extrêmes les plus proches des limites du rectangle sans être à l'intérieur.

3.7 Résultats numériques

Deux versions de l'algorithme proposé, dénommées MSMS1 et MSMS2, ont été implantées en langage C, de même que l'algorithme d'étiquetage de Hansen [36], appelé MSMS0. Les versions MSMS1 et MSMS2 correspondent respectivement à l'initialisation de base et à l'initialisation étendue. Des arbres balancés (AVL) ont été utilisés comme structures de données pour les différents ensembles d'étiquettes.

Les algorithmes ont été testés sur des réseaux aléatoires dans lesquels tout sommet d'indice i est connecté à celui d'indice $i+1$, et aléatoirement à tous les autres, de sorte qu'aucun arc n'arrive au sommet v_1 (source) ni ne part du sommet v_n (puits). Les valeurs de x et de y sont générées indépendamment, de manière aléatoire, pour chaque arc. Ces valeurs sont comprises entre 0 et une borne supérieure variable. Ainsi, l'arc joignant deux sommets d'indices consécutifs n'est pas nécessairement le plus court chemin entre ces sommets, pour l'un ou l'autre des deux critères.

Les caractéristiques des différents problèmes testés sont regroupées dans les tableaux 3.1, 3.7, 3.3 et 3.4. Les trois premiers tableaux traitent respectivement de l'impact de la variation marginale des paramètres suivants: la densité du réseau, le nombre de sommets et l'étendue des critères (i.e., la différence entre la plus grande et la plus petite valeur du critère).

Dans chacun des trois premiers tableaux, la valeur du paramètre correspondant augmente progressivement pour les 10 problèmes considérés, tandis que les deux autres paramètres sont maintenus constants. Les tableaux montrent également le nombre d'étiquettes efficaces extrêmes et non extrêmes dans chaque réseau, ainsi que le temps CPU requis par chacun des algorithmes MSMS0 et MSMS1.

Tableau 3.1 – *Plus courts chemins bicritères pour réseaux de densité croissante*

Caractéristiques du réseau						Temps CPU (sec.)	
Nb. de sommets	Nb. d'arcs	Densité	Étendue des coûts	Nb. d'étiquet. effic. extr.	Nb. d'étiquet. effic. non extr.	MSMS1	MSMS0
300	391	0.004	199.738	2	0	0.180	0.030
300	9203	0.103	199.994	7	8	1.340	2.020
300	18240	0.203	200.000	4	3	1.850	4.740
300	26956	0.301	199.994	5	1	1.810	6.250
300	36005	0.401	200.000	9	15	6.070	12.050
300	45143	0.503	199.994	11	13	8.390	15.940
300	53495	0.596	200.000	11	8	6.220	18.330
300	62719	0.699	200.000	14	13	12.360	23.760
300	71447	0.797	200.000	7	7	7.310	24.910
300	80432	0.897	200.000	11	13	11.720	29.620

Le quatrième tableau donne des statistiques pour 100 réseaux dans lesquels les trois paramètres sont tous aléatoires en même temps. En plus des caractéristiques des réseaux, ce tableau montre les ratios des temps CPU entre les algorithmes MSMS0, MSMS1 et MSMS2, de même que ceux correspondant à la fraction du temps requis par l'initialisation de chacune des deux versions de l'algorithme proposé. Pour chaque indicateur considéré, le tableau donne la moyenne et l'écart-type pour les 100 problèmes générés, de même que les valeurs minimales et maximales observées.

Les tests ont été effectués sur une station de travail Sun Ultra 2. On note une domination presque systématique de l'algorithme proposé (MSMS1) sur celui qui résout le problème par une seule extrémité (MSMS0), en particulier lorsque la densité ou la taille du réseau augmente (tableaux 3.1 et 3.7). La tendance est cependant moins marquée lorsque l'étendue des coûts sur les arcs varie (tableau 3.3).

Lorsque tous les trois paramètres varient aléatoirement, la version MSMS1 est, en moyenne, 2.1 fois plus rapide que l'algorithme MSMS0 et 2.8 fois plus rapide que MSMS2. Toutefois, MSMS0 ne prend qu'environ 83% du temps requis par MSMS2.

Tableau 3.2 – *Plus courts chemins bicritères pour réseaux de taille croissante*

Caractéristiques du réseau						Temps CPU (sec.)	
Nb. de sommets	Nb. d'arcs	Densité	Étendue des coûts	Nb. d'étiquet. effic. extr.	Nb. d'étiquet. effic. non extr.	MSMS1	MSMS0
500	3037	0.012	199.951	4	0	0.570	0.410
1000	10866	0.011	199.988	6	2	1.880	2.350
1500	24152	0.011	199.982	6	6	5.920	6.720
2000	41840	0.010	200.000	5	4	10.020	11.490
2500	65321	0.010	200.000	5	13	18.830	19.530
3000	92739	0.010	200.000	12	5	26.220	32.930
3500	126026	0.010	200.000	4	10	37.470	46.420
4000	164431	0.010	200.000	11	9	52.490	61.830
4500	206932	0.010	200.000	8	10	63.500	83.030
5000	254962	0.010	200.000	10	12	80.870	113.840

On remarque également un pourcentage de temps relativement élevé pour l'étape d'initialisation durant la résolution du problème par les deux extrémités, i.e., en moyenne 45% pour MSMS1 et 68% pour MSMS2. Ce ratio est particulièrement élevé pour MSMS1, puisqu'au plus deux étiquettes efficaces correspondant à des chemins v_1-v_n peuvent être trouvées durant l'initialisation de base.

Cette observation suggère cependant une grande efficacité des tests utilisés durant les phases "en avant" et "en arrière" de l'étape principale de l'algorithme, comparativement à la recherche de plus courts chemins paramétriques. De même, cela explique la moins bonne performance de MSMS2, où toutes les étiquettes efficaces extrêmes sont déterminées pendant l'initialisation, en résolvant plusieurs sous-problèmes de plus courts chemins paramétriques.

Considérons maintenant l'impact du nombre d'étiquettes efficaces sur la performance des algorithmes étudiés. L'algorithme MSMS0 semble meilleur dans les cas (rares) de réseaux de petite taille ou de très faible densité, pour lesquels toutes les étiquettes efficaces sont extrêmes (première ligne des tableaux 3.1 et 3.7).

Tableau 3.3 – *Plus courts chemins bicritères pour étendues de coûts croissantes*

Caractéristiques du réseau						Temps CPU (sec.)	
Nb. de sommets	Nb. d'arcs	Densité	Étendue des coûts	Nb. d'étiquet. effic. extr.	Nb. d'étiquet. effic. non extr.	MSMS1	MSMS0
2000	42082	0.011	99.997	7	4	8.320	14.050
2000	41909	0.010	200.000	7	7	10.940	12.940
2000	42250	0.011	300.000	8	10	11.380	12.700
2000	41992	0.011	400.000	6	4	10.330	11.780
2000	41799	0.010	500.000	4	5	10.450	11.610
2000	41905	0.010	600.000	6	9	11.760	13.320
2000	42070	0.011	700.000	5	9	10.810	14.620
2000	42044	0.011	800.000	6	7	12.010	12.330
2000	41772	0.010	900.000	7	3	11.090	12.540
2000	42270	0.011	1000.000	7	5	11.110	15.620

Ceci peut s'expliquer par l'inefficacité relative de la recherche des plus courts chemins paramétriques. Cependant, la méthode de résolution par les deux extrémités semble présenter une plus grande robustesse lorsque la densité du réseau varie significativement mais que le nombre total d'étiquettes efficaces change peu (tableau 3.1).

Ces résultats suggèrent que l'algorithme proposé, dans sa version de base, a une meilleure performance que l'algorithme d'étiquetage à partir de la source, lorsque la taille ou la densité du graphe augmente. Le gain de vitesse reste cependant faible (environ 2 en moyenne) mais se confirme sur une grande variété de problèmes sans aucune corrélation particulières entre les valeurs de x et de y. Des résultats semblables ont également été obtenus en utilisant uniquement des valeurs entières pour les deux critères.

Notons que ces observations, faites sur des réseaux aléatoires, peuvent difficilement être extrapolées. Cependant, on peut raisonnablement s'attendre à une certaine supériorité de l'approche par les deux extrémités pour d'autres types de problèmes, incluant les cas pratiques de grande taille.

Tableau 3.4 – *Plus courts chemins bicritères pour réseaux aléatoires*

	Moyenne	Écart-type	Min.	Max.
Nb. de sommets	205.63	60.63	104	297
Nb. d'arcs	24857.55	24857.31	828	83736
Densité	0.535	0.289	0.016	0.985
Étendue des coûts	100.818	57.462	0.476	198.169
Nb. d'étiquet. effic. extr.	6.55	2.20	1	13
Nb. d'étiquet. effic. non extr.	8.10	4.56	0	22
MSMS0 / MSMS1	2.135	0.731	0.333	4.828
MSMS0 / MSMS2	0.833	0.512	0.121	4.382
MSMS2 / MSMS1	2.801	0.801	0.986	5.541
Initialisation / MSMS1	0.455	0.134	0.196	0.971
Initialisation / MSMS2	0.683	0.132	0.000	0.841

Cette remarque se justifie par le fait que l'algorithme proposé est aussi une méthode d'étiquetage, mais exploite de l'information en provenance des deux extrémités, afin de réduire le nombre total d'étiquettes temporairement efficaces qui sont prolongées. Les tests développés à ce effet semblent relativement efficaces, même si la phase d'initialisation, où l'algorithme de Dijkstra [25] est exécuté quatre fois, l'est moins.

Le problème de plus court chemin bicritère étudié dans ce chapitre implique la minimisation des deux critères considérés. Dans certaines applications pratiques, cependant, des bornes peuvent être imposées sur la valeur de chacun des deux critères, afin d'éviter les solutions impliquant une trop grande détérioration de l'un d'eux (voir la section 3.6).

Cette considération nous amène à un problème voisin, où l'on considère plusieurs critères mais dont un seul est minimisé et des bornes, inférieures et supérieures, sont définies pour tous les autres, sur chaque arc du réseau. Le chapitre suivant présente quelques algorithmes pour ce problème.

CHAPITRE 4

ALGORITHMES DE PLUS COURT CHEMIN AVEC FENÊTRES DE RESSOURCE

Nous étudions, dans ce chapitre, la résolution du problème de plus court chemin avec fenêtres de ressource, RCSPP, introduit à la section 1.3. Des algorithmes pseudo-polynomiaux, basés sur la programmation dynamique et sur une approche en deux phases, sont proposés pour le problème. L'algorithme en deux phases permet de traiter efficacement les cas de réoptimisation, lorsque certains sommets sont supprimés ou sont fixés, ou lorsque les coûts changent. Nous définissons ci-après quelques notations additionnelles qui seront utilisées dans l'évaluation de la complexité des différents algorithmes.

4.1 Notations supplémentaires

Les notations définies ici s'ajoutent à celles introduites à la section 1.3. Étant donnée une ressource r, considérons, pour chaque sommet $v_i \in V$, les bornes inférieures et supérieures suivantes sur la r-ième composante du vecteur de ressources, x_i, pour tout chemin v_1-v_i réalisable:

$$\underline{\varphi}_{ir} = \min\{\underline{h}_{ir}, \underline{x}_{ir}, \overline{x}_{ir}\} \quad \text{et} \quad \overline{\varphi}_{ir} = \max\{\overline{h}_{ir}, \overline{x}_{ir}, \underline{x}_{ir}\}.$$

Il est clair, à partir de la définition de l'opérateur de mise à jour $\Psi(\cdot,\cdot)$ (section 1.3.1), que $\underline{\varphi}_{ir} \leq x_{ir} \leq \overline{\varphi}_{ir}$. Soient maintenant les paramètres ci-après:

$$\varphi = \sum_{(v_i,v_j)\in A} \varphi_i, \quad \text{où} \quad \varphi_i = \prod_{r\in\mathcal{R}} (\overline{\varphi}_{ir} - \underline{\varphi}_{ir} + 1),$$

et

$$\theta = \sum_{(v_i,v_j)\in A} \theta_{ij}, \quad \text{où} \quad \theta_{ij} = \prod_{r\in\mathcal{R}} (\min\{\overline{\varphi}_{ir}, \overline{w}_{ijr}\} - \max\{\underline{\varphi}_{ir}, \underline{w}_{ijr}\} + 1).$$

Pour tout sommet $v_i \in V$, le paramètre φ_i est une limite supérieure sur le nombre de vecteurs de consommations de ressource pour les chemins v_1-v_i réalisables, tandis que φ est une borne supérieure sur le nombre total de vecteurs de consommations de ressource pour l'ensemble du graphe. De façon similaire, θ_{ij} borne le nombre de vecteurs de consommations de ressource pour les chemins v_1-v_j réalisables qui passent par un arc (v_i, v_j) donné et θ est une limite sur le nombre total de ces vecteurs pour tout le graphe. On peut remarquer que $\theta \leq \varphi$, puisque $\theta_{ij} \leq \varphi_i$, pour tout arc $(v_i, v_j) \in A$. On a également $\theta \leq \sum_{(v_i,v_j)\in A} \prod_{r\in\mathcal{R}} (\overline{w}_{ijr} - \underline{w}_{ijr} + 1)$, de manière similaire.

Supposons que les vecteurs de consommations de ressource à un sommet donné $v_i \in V$ sont disposés par un ordre lexicographique dans une structure d'arbre où il correspondent aux feuilles. Les niveaux de l'arbre correspondent aux différentes ressources dans l'ensemble $\mathcal{R}$. Un noeud du r-ième niveau a au plus $O(\overline{\varphi}_{ir} - \underline{\varphi}_{ir} + 1)$ enfants et peut, par conséquent, tenir dans un tableau unidimensionnel de taille $O(\overline{\varphi}_{ir} - \underline{\varphi}_{ir} + 1)$. La position d'une valeur x_{ir} quelconque est alors donnée par $x_{ir} - \underline{\varphi}_{ir} + 1$. Ainsi, un fils donné de ce noeud peut être localisé en temps constant, et par conséquent la feuille correspondant à un vecteur de ressource $x_i = (x_{i1}, x_{i2}, ..., x_{i|\mathcal{R}|})$ peut être localisée en $O(|\mathcal{R}|)$ opérations.

D'une manière similaire, la position lexicographique du vecteur de ressource d'un chemin v_1-v_i pouvant être prolongé par un arc $(v_i, v_j) \in A$ donné, peut être trouvée en temps $O(|\mathcal{R}|)$. La taille d'un noeud dans l'arbre correspondant est de l'ordre de $O(\min\{\overline{\varphi}_{ir}, \overline{w}_{ijr}\} - \max\{\underline{\varphi}_{ir}, \underline{w}_{ijr}\} + 1)$.

4.2 Une approche par le graphe des états

Une première méthode de résolution consiste à générer explicitement le graphe $G_E = (S, E)$ de tous les états possibles des sous-chemins réalisables en provenance de la source, et à exécuter ensuite un algorithme classique de plus court chemin sur ce graphe élargi. Étant donné $v_i \in V$, un sommet $s_i \in S$ peut être associé à chaque vecteur de ressource x_i correspondant à un chemin v_1-v_i. L'ensemble de tous les sommets $s_i \in S$, qui sont associés à un sommet $v_i \in V$ donné, sera noté S_i. L'arc (s_i, s_j) est dans E si et seulement si $\underline{w}_{ij} \leq x_i \leq \overline{w}_{ij}$ et $x_j = \Psi(v_j, x_i + u_{ij})$, où x_i et x_j sont les vecteurs de consommations de ressource associés à s_i et s_j respectivement.

L'ensemble de tous les arcs $(s_i, s_j) \in E$, associés à un arc $(v_i, v_j) \in A$ donné, sera noté E_{ij}. Le coût sur l'arc (s_i, s_j) est égal à c_{ij}. Un sommet supplémentaire s_{n+1} (puits) est ajouté à S, et des arcs (s_n, s_{n+1}), pour $s_n \in S_n$, sont ajoutés à E. Le graphe résultant G_E est un graphe orienté qui peut être généré par la récurrence ci-après, suivant un ordre croissant de l'indice de v_j où x_0 est le vecteur de ressource initial.

$$\begin{aligned}
S_1 &= \{s_1 : x_1 = \Psi(v_1, x_0)\}, \\
S_j &= \{s_j : x_j = \Psi(v_j, x_i + u_{ij}),\ \underline{w}_{ij} \leq x_i \leq \overline{w}_{ij},\ s_i \in S_i,\ (v_i, v_j) \in A\}, \\
E_{ij} &= \{(s_i, s_j) : x_j = \Psi(v_j, x_i + u_{ij}),\ \underline{w}_{ij} \leq x_i \leq \overline{w}_{ij},\ s_i \in S_i,\ s_j \in S_j\}, \\
E_{nn+1} &= \{(s_n, s_{n+1}) : s_n \in S_n\}.
\end{aligned}$$

La longueur du chemin optimal est alors donnée par la récurrence suivante, par indice croissant de s_j, tandis que la complexité de l'ensemble du processus est donnée par le théorème 4.1.

$$c_1 = 0, \quad c_j = \min\{c_i + c_{ij} : (s_i, s_j) \in E\}.$$

Théorème 4.1 *L'approche par le graphe des états résout RCSPP en temps $O(\varphi|\mathcal{R}|)$.*

Preuve. Le graphe G_E peut être généré en temps $O(\varphi|\mathcal{R}|)$ si, pour chaque $v_i \in V$, les éléments s_i de S_i sont disposés par ordre lexicographique des vecteurs de ressource x_i associés. En effet, pour tout $v_i \in V$, S_i contient au plus $O(\varphi_i)$ éléments (voir la

section 4.1) et S_1 peut être déterminé en temps $O(|\mathcal{R}|)$. Soit un arc $(v_i, v_j) \in A$ tel que S_i est connu. Chaque arc $(s_i, s_j) \in E_{ij}$ est déterminé en sélectionnant un élément $s_i \in S_i$ puis en testant si le vecteur x_i associé satisfait $\underline{w}_{ij} \leq x_i \leq \overline{w}_{ij}$, avant de calculer, si nécessaire, $x_j = \Psi(v_j, x_i + u_{ij})$. Ceci peut se faire en $O(|\mathcal{R}|)$ opérations, en traitant les éléments de S_i consécutivement. L'insertion de s_j dans S_j requiert $O(|\mathcal{R}|)$ opérations pour vérifier si une copie existe déjà (voir la section 4.1).

Par conséquent chaque arc $(v_i, v_j) \in A$ peut être examiné en $O(\varphi_i|\mathcal{R}|)$ opérations et les arcs $(s_n, s_{n+1}) \in E$ sont crées en temps $O(\varphi_n)$. La complexité pour la génération de G_E est donc en $O(\varphi|\mathcal{R}|)$ si les sommets v_j sont considérés dans l'ordre croissant de leurs indices.

En outre, étant donné un arc $(v_i, v_j) \in A$, chaque sommet $s_i \in S_i$ est connecté à, au plus, un sommet $s_j \in S_j$, puisque $\Psi(v_j, x_i + u_{ij})$ est unique et x_i pourrait ne pas satisfaire $\underline{w}_{ij} \leq x_i \leq \overline{w}_{ij}$. Il y a donc $O(\varphi)$ arcs dans E, dont chacun peut être examiné en temps constant, durant la récurrence qui donne le chemin optimal. Ceci permet de calculer les valeurs de c_j et de déterminer un pointeur sur le prédécesseur de s_j le long du chemin optimal, i.e., sur s_i tel que $c_j = c_i + c_{ij}$. La complexité de l'approche par le graphe des états est, par conséquent, celle de la génération de G_E. □

4.3 Une approche de programmation dynamique

Soient x_0 le vecteur de ressource initial et, pour chaque sommet $v_j \in V$, l'ensemble F_j de toutes les paires formées par les longueurs de chemin v_1-v_j réalisables et les vecteurs de consommations de ressource associés. Considérons la récurrence suivante, par indice croissant des v_j:

$$\begin{aligned} F_1 &= \{(x_1, 0) : x_1 = \Psi(v_1, x_0)\}, \\ F_j &= \{(x_j, c_j) : c_j = \min\{c_i + c_{ij} : \underline{w}_{ij} \leq x_i \leq \overline{w}_{ij},\ x_j = \Psi(v_j, x_i + u_{ij}), \\ &\qquad\qquad (x_i, c_i) \in F_i,\ (v_i, v_j) \in A\}\}. \end{aligned}$$

La longueur du chemin optimal est alors donnée par: $\min\{c_n : (x_n, c_n) \in F_n\}$.

Notons qu'étant donné un sommet v_i, l'espace mémoire occupé par l'ensemble F_i peut être libéré si tous les ensembles F_j tels que $(v_i, v_j) \in A$ sont déterminés. Le résultat suivant donne la complexité de la méthode.

Théorème 4.2 *L'approche de la programmation dynamique détermine le chemin optimal pour RCSPP en temps* $O(\varphi|\mathcal{R}|)$.

Preuve. Les sommets sont considérés par ordre croissant des indices. L'exactitude de cette récurrence "en avant" de programmation dynamique vient du fait que chaque paire $(x_j, c_j) \in F_j$ correspond à un plus court chemin v_1-v_j réalisable. Tous les vecteurs de ressource pouvant être obtenus à partir des sommets v_i tels que $(v_i, v_j) \in A$ sont considérés lors du calcul de x_j. Par conséquent, une seule valeur de c_j est associée à chaque vecteur x_j et F_j contient au plus φ_j paires (x_j, c_j).

Étant donnés $(v_i, v_j) \in A$ et $(x_i, c_i) \in F_i$, vérifier si $\underline{w}_{ij} \leq x_i \leq \overline{w}_{ij}$ et calculer $x_j = \Psi(v_j, x_i + u_{ij})$ et $c_j = c_i + c_{ij}$ requiert $O(|\mathcal{R}|)$ opérations. Vérifier si (x_j, c_j) doit remplacer la paire (x_j, c_j^*) courante de F_j dont la valeur c_j^* est minimale pour le même vecteur x_j requiert $O(|\mathcal{R}|)$ opérations si les éléments de F_j sont classés par ordre lexicographique des vecteurs de ressource associés (voir la section 4.1). Ainsi, le calcul des ensembles F_j nécessite $O(\varphi|\mathcal{R}|)$ opérations en tout si les éléments de F_j, pour $v_j \in V$, sont examinés consécutivement. Finalement, la sélection du chemin optimal se fait en temps $O(\varphi_n)$.□

On remarque qu'à un sommet donné $v_i \in V$, l'ensemble F_i peut contenir plusieurs paires (x_i, c_i) ne satisfaisant, sur aucun arc $(v_i, v_j) \in A$, les contraintes de fenêtres de ressource (1.3). On pourrait donc envisager une récurrence en "en arrière" dans le but d'éviter la détermination de ces paires, et ainsi réduire le temps de calcul. Cependant, certaines difficultés techniques surgissent, du fait notamment des opérations de mise à jour. En effet, étant donné un vecteur de ressource x_j au sommet v_j et un arc $(v_i, v_j) \in A$, plusieurs vecteurs de ressource x_i correspondant à différents chemins v_1-v_i réalisables peuvent être solutions de l'équation $x_j = \Psi(v_j, x_i + u_{ij})$. S'il n'y avait pas de mise à jour, seul le calcul de $x_i = x_j - u_{ij}$ serait nécessaire.

Notons également que le facteur φ dans la complexité de la récurrence de programmation dynamique et de celle utilisant le graphe des états, implique une charge de calcul considérable. Ceci est particulièrement vrai lorsque le problème de plus court chemin avec fenêtres de ressource doit être résolu de manière répétitive, pour différentes valeurs de coûts sur les arcs mais avec les mêmes consommations de ressource. Cette situation se rencontre notamment lorsque le problème apparaît comme problème auxiliaire durant un processus de génération de colonnes. Nous introduisons, à la section prochaine, une procédure en deux phases utilisant une récurrence "en arrière" et nécessitant moins de calculs dans de tels cas de réoptimisation.

4.4 Un algorithme en deux phases

4.4.1 Bornes sur les consommations de ressource

Nous décrivons d'abord une procédure pour calculer de meilleures valeurs pour les bornes inférieures et supérieures, $\underline{\varphi}_{jr}$ et $\overline{\varphi}_{jr}$, sur la consommation de la ressource r pour les chemins v_1-v_j réalisables. Considérons, pour chaque sommet $v_j \in V$ et chaque ressource $r \in \mathcal{R}$, la récurrence "en avant", i.e., par ordre croissant des indices de v_j, suivante:

$$\underline{\varphi}_{1r} = \min\{\underline{h}_{1r}, \underline{x}_{1r}, \overline{x}_{1r}\}, \qquad \overline{\varphi}_{1r} = \max\{\overline{h}_{1r}, \overline{x}_{1r}, \underline{x}_{1r}\}, \tag{4.1}$$

$$\underline{\varphi}_{jr} = \min\{\underline{\varphi}_{ijr} : (v_i, v_j) \in A\}, \qquad \overline{\varphi}_{jr} = \max\{\overline{\varphi}_{ijr} : (v_i, v_j) \in A\}, \tag{4.2}$$

où:

$$\underline{\varphi}_{ijr} = \begin{cases} \min\{\overline{x}_{jr}, \max\{\underline{\varphi}_{ir}, \underline{w}_{ijr}\} + u_{ijr}\} & \text{si } \underline{h}_{jr} \le \max\{\underline{\varphi}_{ir}, \underline{w}_{ijr}\} + u_{ijr} \le \overline{h}_{jr}, \\ \min\{\underline{h}_{jr}, \underline{x}_{jr}, \overline{x}_{jr}\} & \text{sinon,} \end{cases} \tag{4.3}$$

$$\overline{\varphi}_{ijr} = \begin{cases} \max\{\underline{x}_{jr}, \min\{\overline{\varphi}_{ir}, \overline{w}_{ijr}\} + u_{ijr}\} & \text{si } \underline{h}_{jr} \le \min\{\overline{\varphi}_{ir}, \overline{w}_{ijr}\} + u_{ijr} \le \overline{h}_{jr}, \\ \max\{\overline{h}_{jr}, \overline{x}_{jr}, \underline{x}_{jr}\} & \text{sinon.} \end{cases} \tag{4.4}$$

Les équations (4.1) doivent être remplacées par $\underline{\varphi}_{1r} = \overline{\varphi}_{1r} = \Psi(v_1, x_{0r})$ si RCSPP est

résolu pour un seul vecteur initial x_0. La proposition 4.4.1 donne une propriété des paramètres ainsi calculés.

Proposition 4.4.1 *Soient un chemin v_1-v_j réalisable et x_j son vecteur de consommations de ressource au sommet v_j, alors $\underline{\varphi}_{jr} \leq x_{jr} \leq \overline{\varphi}_{jr}$, pour $r \in \mathcal{R}$. En plus, les bornes $\underline{\varphi}_{jr}$ et $\overline{\varphi}_{jr}$, pour $r \in \mathcal{R}$ et $v_j \in V$, peuvent être calculées en $O(|\mathcal{R}|\, m)$ opérations.*

Preuve. La première partie de la propriété sera prouvée par récurrence. Soit x_0 le vecteur de ressource initial du chemin. Puisque, pour tout $r \in \mathcal{R}$, $x_{1r} = \Psi(v_1, x_{0r})$, on a, par définition de $\Psi(\cdot,\cdot)$: $x_{1r} = x_{0r}$ si $\underline{h}_{1r} \leq x_{0r} \leq \overline{h}_{1r}$, et $x_{1r} = \underline{x}_{1r}$ ou $x_{1r} = \overline{x}_{1r}$ sinon. Par conséquent $\underline{\varphi}_{1r} \leq x_{1r} \leq \overline{\varphi}_{1r}$.

Soit maintenant (v_i, v_j) le dernier arc du chemin et supposons, par récurrence, que $\underline{\varphi}_{ir} \leq x_{ir} \leq \overline{\varphi}_{ir}$. Puisque le chemin v_1-v_j est réalisable, nous avons également $\underline{w}_{ijr} \leq x_{ir} \leq \overline{w}_{ijr}$ et, par suite, $\max\{\underline{\varphi}_{ir}, \underline{w}_{ijr}\} \leq x_{ir} \leq \min\{\overline{\varphi}_{ir}, \overline{w}_{ijr}\}$. En outre, puisque $x_{jr} = \Psi(v_j, x_{ir} + u_{ijr})$, nous avons: $x_{jr} = x_{ir} + u_{ijr}$ si $\underline{h}_{jr} \leq x_{ir} + u_{ijr} \leq \overline{h}_{jr}$ et $x_{jr} = \underline{x}_{jr}$ ou $x_{jr} = \overline{x}_{jr}$ sinon. On en déduit donc:

- si $\underline{h}_{jr} \leq \max\{\underline{\varphi}_{ir}, \underline{w}_{ijr}\} + u_{ijr} \leq \overline{h}_{jr}$ alors $\max\{\underline{\varphi}_{ir}, \underline{w}_{ijr}\} + u_{ijr} \leq x_{jr}$ si $x_{ir} + u_{ijr} \leq \overline{h}_{jr}$, et $x_{jr} = \overline{x}_{jr}$ sinon; donc $\underline{\varphi}_{ijr} \leq x_{jr}$;
- Si la condition $\underline{h}_{jr} \leq \max\{\underline{\varphi}_{ir}, \underline{w}_{ijr}\} + u_{ijr} \leq \overline{h}_{jr}$ n'est pas respectée, alors $x_{jr} = \underline{x}_{jr}$ si $x_{ir} + u_{ijr} < \underline{h}_{jr}$, $\underline{h}_{jr} \leq x_{jr}$ si $\underline{h}_{jr} \leq x_{ir} + u_{ijr} \leq \overline{h}_{jr}$, et $x_{jr} = \overline{x}_{jr}$ si $\overline{h}_{jr} < x_{ir} + u_{ijr}$; par conséquent $\underline{\varphi}_{ijr} \leq x_{jr}$;
- Si $\underline{h}_{jr} \leq \min\{\overline{\varphi}_{ir}, \overline{w}_{ijr}\} + u_{ijr} \leq \overline{h}_{jr}$ alors $x_{jr} \leq \min\{\overline{\varphi}_{ir}, \overline{w}_{ijr}\} + u_{ijr}$ if $\underline{h}_{jr} \leq x_{ir} + u_{ijr}$, et $x_{jr} = \underline{x}_{jr}$ sinon; d'où $x_{jr} \leq \overline{\varphi}_{ijr}$;
- Si la condition $\underline{h}_{jr} \leq \min\{\overline{\varphi}_{ir}, \overline{w}_{ijr}\} + u_{ijr} \leq \overline{h}_{jr}$ n'est pas vérifiée, alors $x_{jr} = \underline{x}_{jr}$ si $x_{ir} + u_{ijr} < \underline{h}_{jr}$, $x_{jr} \leq \overline{h}_{jr}$ si $\underline{h}_{jr} \leq x_{ir} + u_{ijr} \leq \overline{h}_{jr}$, et $x_{jr} = \overline{x}_{jr}$ si $\overline{h}_{jr} < x_{ir} + u_{ijr}$; par conséquent $x_{jr} \leq \overline{\varphi}_{ijr}$.

Ainsi, $\underline{\varphi}_{jr} \leq x_{jr} \leq \overline{\varphi}_{jr}$. En ce qui concerne la complexité, étant donné $r \in \mathcal{R}$, le calcul de $\underline{\varphi}_{1r}$ et de $\overline{\varphi}_{1r}$, requiert un temps constant, de même que le calcul de $\underline{\varphi}_{ijr}$ et de $\overline{\varphi}_{ijr}$,

pour chaque arc $(v_i, v_j) \in A$ si $\underline{\varphi}_{ir}$ et $\overline{\varphi}_{ir}$ sont connus. Les valeurs $\underline{\varphi}_{jr}$ et $\overline{\varphi}_{jr}$ peuvent être calculées en un temps proportionnel au nombre d' arcs $(v_i, v_j) \in A$. Donc, la récurrence requiert un temps $O(|\mathcal{R}|\, m)$ en tout si les sommets $v_j \in V$ sont considérés dans l'ordre croissant de leurs indices, puisque, pour chacun de ces sommets v_j, seuls les arcs $(v_i, v_j) \in A$ ont besoin d'être examinés. □

4.4.2 Caractérisation des consommations de ressource

L'algorithme utilise, dans la phase 1, une récurrence "en arrière" pour caractériser la consommation de ressource des sous-chemins dont le prolongement peut donner un chemin v_1-v_n réalisable. Ceci se fait en calculant, pour chaque arc $(v_i, v_j) \in A$, un ensemble U_{ij}, par ordre décroissant des indices des sommets v_i (voir la figure 4.1) :

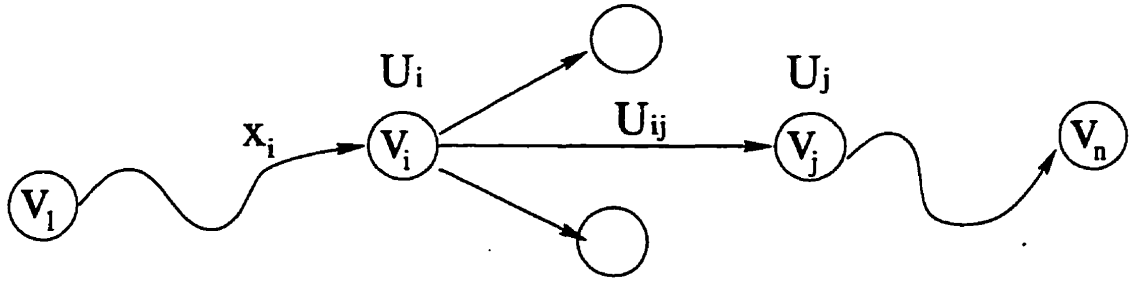


Figure 4.1 – *Calcul des étiquettes de chemins v_i-v_n réalisables.*

$$U_{ij} = \{x_i \in \mathbf{Z}^{|\mathcal{R}|} : \max\{\underline{\varphi}_{ir}, \underline{w}_{ijr}\} \leq x_{ir} \leq \min\{\overline{\varphi}_{ir}, \overline{w}_{ijr}\},\ r \in \mathcal{R},$$
$$\Psi(v_j, x_i + u_{ij}) \in U_j \text{ si } v_j \neq v_n, \}.$$
$$U_i = \bigcup_{(v_i, v_j) \in A} U_{ij}.$$

Nous montrons maintenant que U_{ij} contient effectivement les vecteurs de ressource des chemins v_1-v_i dont le prolongement, par l'arc arc (v_i, v_j), peut donner un chemin v_1-v_n réalisable.

Proposition 4.4.2 *Soit un arc $(v_i, v_j) \in A$. Un chemin v_1-v_i réalisable, avec un vecteur de ressource x_i en v_i, peut donner un chemin v_1-v_n réalisable et passant par l'arc (v_i, v_j) si et seulement si $x_i \in U_{ij}$.*

Preuve. Considérons la partie "si", i.e. la conditions suffisante. Par construction, $x_i \in U_{ij}$ implique $\underline{w}_{ijr} \leq x_{ir} \leq \overline{w}_{ijr}$, $r \in \mathcal{R}$. En plus, $\Psi(v_j, x_i + u_{ij}) \in U_j$, si $v_j \neq v_n$. On en déduit un chemin v_1-v_j réalisable dont le vecteur de consommation de ressource, $x_j = \Psi(v_j, x_i + u_{ij})$, est dans U_j si $v_j \neq v_n$. Il s'en suit donc qu'il existe un sommet $v_k \in V$ tel que $(v_j, v_k) \in A$ et $x_k \in U_{jk}$. Puisque le réseau est acyclique et le nombre d'arcs est fini, le résultat suit.

Pour la condition nécessaire, supposons (v_i, v_n) est le dernier arc du chemin v_1-v_n réalisable résultant. Alors, pour $r \in \mathcal{R}$, $\underline{w}_{inr} \leq x_{ir} \leq \overline{w}_{inr}$ et, puisque le sous-chemin v_1-v_i extrait est aussi réalisable, nous avons, par la proposition 4.4.1, $\underline{\varphi}_{ir} \leq x_{ir} \leq \overline{\varphi}_{ir}$. D'où $x_i \in U_{in}$ et $x_i \in U_i$. En plus, considérons un arc quelconque (v_i, v_j) du chemin v_1-v_n, tel que $v_j \neq v_n$ et supposons, par récurrence, que $x_j \in U_j$. Puisque les sous-chemins v_1-v_i et v_1-v_j extraits sont aussi réalisables, nous avons, pour $r \in \mathcal{R}$, $\underline{w}_{ijr} \leq x_{ir} \leq \overline{w}_{ijr}$ et $\underline{\varphi}_{ir} \leq x_{ir} \leq \overline{\varphi}_{ir}$, par la proposition 4.4.1, de même que $x_j = \Psi(v_j, x_i + u_{ij})$. Donc $x_i \in U_{ij}$. □

Notons qu'étant donné un sommet v_j la mémoire allouée pour U_j peut être libérée dès que tous les arcs arrivant à v_j sont examinés. On peut aussi remarquer que pour un arc $(v_i, v_j) \in A$ donné, ce ne sont pas tous les vecteurs dans U_{ij} qui correspondent effectivement à des chemins v_1-v_i.

Pour chaque sommet v_j nous décrivons un processus pour sélectionner, dans tous les ensembles U_{ij} associés à v_j, les vecteurs correspondant aux chemins v_1-v_n qui passent par v_j. L'ensemble de ces vecteurs sera noté D_j. Pour chaque vecteur $x_j \in D_j$, l'ensemble $R(x_j)$ des vecteurs de consommations de ressource x_i (aux prédécesseurs v_i de v_j) desquels x_j peut être obtenu, est aussi déterminé durant le calcul des D_j. Puisque les éléments de U_{ij} sont visités séquentiellement, il n'est pas nécessaire de les disposer par ordre lexicographique.

La figure 4.2 illustre le calcul des ensembles D_j et $R(x_j)$, pour tout $x_j \in D_j$. Ce calcul se fait par la récurrence suivante, par indice croissant des v_j:

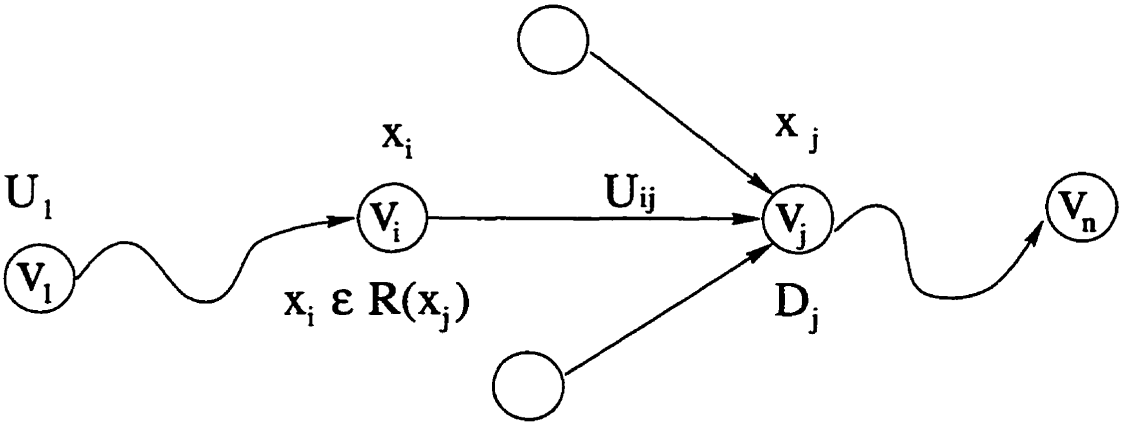


Figure 4.2 – *Calcul des étiquettes de chemins v_1-v_i-v_n réalisables.*

$$
\begin{aligned}
D_1 &= U_1, \\
D_j &= \{x_j : x_j = \Psi(v_j, x_i + u_{ij}),\ x_i \in U_{ij} \cap D_i,\ (v_i, v_j) \in A\}, \\
R(x_j) &= \{x_i : x_j = \Psi(v_j, x_i + u_{ij}),\ x_i \in U_{ij} \cap D_i,\ (v_i, v_j) \in A\}, \quad \forall x_j \in D_j.
\end{aligned}
$$

Nous avons le résultat suivant.

Proposition 4.4.3 *Étant donné un arc $(v_i, v_j) \in A$, il existe un chemin v_1-v_n réalisable passant par (v_i, v_j), avec, pour vecteurs de consommations de ressource, x_i en v_i et x_j en v_j, si et seulement si $x_j \in D_j$ et $x_i \in R(x_j)$.*

Preuve. Pour la partie "si", soient deux vecteurs x_i et x_j tels que $x_j \in D_j$ et $x_i \in R(x_j)$. Il est clair que $x_i \in U_{ij}$ et, par conséquent, un chemin v_1-v_i réalisable ayant comme vecteur de consommations de ressource x_i, s'il en existe, peut être prolongé en un chemin v_1-v_n réalisable. En outre, $x_i \in D_i$ et il existe donc un sommet v_k tel que $(v_k, v_i) \in A$ et $x_k \in R(x_i)$, ce qui implique que $x_k \in D_k$ et $x_k \in U_{ki}$. Puisque le graphe est acyclique, il s'en suit qu'on peut en déduire un chemin v_1-v_i réalisable,

ayant comme premier arc (v_1, v_ℓ) et comme vecteur de ressource initial x_0, tels que $\Psi(v_j, x_0) \in D_1$ et $\Psi(v_j, x_0) \in U_{1\ell}$. D'où le résultat.

Pour voir que la condition est également nécessaire, considérons le premier arc (v_1, v_ℓ) d'un chemin v_1-v_n réalisable passant par l'arc (v_i, v_j) et ayant pour vecteurs de ressource x_i en v_i et x_j en v_j. Soient également x_0, x_1 et x_ℓ, respectivement le vecteur de ressource initial, et les vecteurs de ressource (mis à jour) en v_1 et en v_ℓ. Nous avons $x_1 = \Psi(v_j, x_0)$ et $x_\ell = \Psi(v_\ell, x_1 + u_{1\ell})$. Puisque le chemin est réalisable, on a également $x_1 \in U_{1\ell}$ et, par suite, $x_1 \in D_1$. Supposons maintenant, par récurrence, que $x_i \in D_i$. Le chemin étant réalisable, nous avons, $x_j = \Psi(v_j, x_i + u_{ij})$ et $x_i \in U_{ij}$. Par conséquent, $x_j \in D_j$ et $x_i \in R(x_j)$.□

La complexité du processus complet de caractérisation des consommations de ressource est donnée ci-après.

Proposition 4.4.4 *Le calcul de $R(x_j)$, pour tous les $x_j \in D_j$ et $v_j \in V \backslash \{v_1\}$, peut se faire en temps $O(\theta\,|\mathcal{R}|)$. En outre, le nombre total de vecteurs dans tous les ensembles $R(x_j)$ dans le graphe est borné par θ.*

Preuve. Nous montrons d'abord que les ensembles U_{ij} peuvent être déterminés en temps $O(\theta\,|\mathcal{R}|)$. En effet, le calcul des bornes $\underline{\varphi}_{ir}, \overline{\varphi}_{ir}$, pour $r \in \mathcal{R}$ and $v_i \in V$, requiert $O(|\mathcal{R}|\,m)$ opérations (voir la proposition 4.4.1). Étant donné un arc $(v_i, v_j) \in A$, il y a au plus θ_{ij} vecteurs x_i tels que $\max\{\underline{\varphi}_{ir}, \underline{w}_{ijr}\} \leq x_{ir} \leq \min\{\overline{\varphi}_{ir}, \overline{w}_{ijr}\}$, pour $r \in \mathcal{R}$, ainsi $|U_{ij}| \leq \theta_{ij}$. Pour chacun de ces x_i, le calcul de $\Psi(v_j, x_i + u_{ij})$ requiert un temps $O(|\mathcal{R}|)$, de même que la vérification de $\Psi(v_j, x_i + u_{ij}) \in U_j$, si une représentation lexicographique de U_j est utilisée. La même complexité est nécessaire pour insérer x_i dans U_{ij} et dans U_i par ordre lexicographique. Puisque, pour chaque sommet v_i (considéré par ordre décroissant des indices) seuls les arcs $(v_i, v_j) \in A$ sont examinés, la complexité du calcul des ensembles U_{ij} est $O(\theta\,|\mathcal{R}|)$.

Considérons maintenant la récurrence "en avant", pour la détermination des ensembles D_j et $R(x_j)$, en supposant que U_1 et, pour chaque arc $(v_i, v_j) \in A$, l'ensemble U_{ij} est disponible, de même que $\Psi(v_j, x_i + u_{ij})$, pour tout sommet $x_i \in U_{ij}$ (il

a été calculé lors de la détermination de U_{ij}). D_1 peut donc être déterminé en temps constant.

Étant donné un arc $(v_i, v_j) \in A$, chacun des $O(\theta_{ij})$ vecteurs $x_i \in U_{ij}$ est examiné en vue d'introduire le vecteur $x_j = \Psi(v_j, x_i + u_{ij})$ correspondant dans D_j et x_i dans $R(x_j)$ si nécessaire. Vérifier si chacun de ces vecteurs x_i est dans D_i peut se faire en temps $O(|\mathcal{R}|)$ si les éléments de D_i sont disposés en ordre lexicographique. La même complexité est requise pour introduire x_j dans D_j. Un lien de x_j à x_i peut ensuite être créé (i.e. x_i est introduit dans $R(x_j)$) en temps constant. Si x_j existe déjà dans D_j, un lien est créé entre la copie existante et le vecteur x_i au sommet v_i.

Ainsi, la récurrence "en avant" requiert un temps $O(\theta\,|\mathcal{R}|)$ en tout, tandis qu'au plus θ_{ij} liens sont créés lorsque chaque arc $(v_i, v_j) \in A$ est examiné. Le résultat est dès lors immédiat. □

4.4.3 Recherche d'un chemin optimal

La phase 2 de l'algorithme est illustrée à la figure 4.3 et correspond à la récurrence ci-après, suivant l'ordre croissant de l'indice des $v_j \in V\backslash\{v_1\}$, où x_1 est posé égal à $\Psi(v_1, x_0)$:

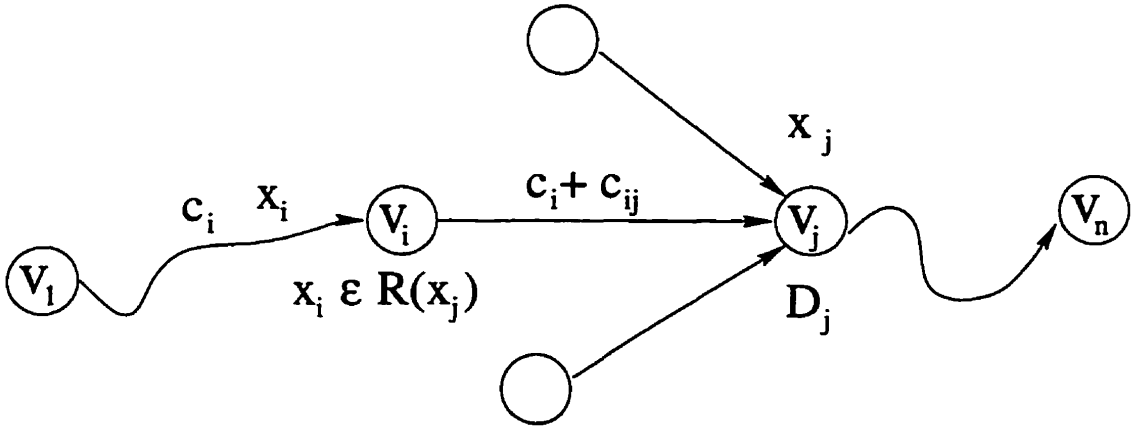


Figure 4.3 – *Calcul du chemin optimal*

$$
\begin{aligned}
c(x_1) &= 0 \quad \text{si } x_1 \in D_1. \quad \text{Sinon FIN.} \\
c(x_1') &= +\infty \quad \text{pour tout } x_1' \in D_1 \backslash \{x_1\}. \\
c(x_j) &= c(x_{i^\bullet}) + c_{i^\bullet j} \quad \text{et} \quad p(x_j) = v_{i^\bullet}, \quad \text{pour tout} \quad x_j \in D_j,\ v_j \in V \backslash \{v_1\}, \\
&\text{où } v_{i^\bullet} \text{ est tel que: } x_{i^\bullet} = \operatorname{argmin}\{c(x_i) + c_{ij} : x_i \in R(x_j)\}.
\end{aligned}
$$

Le chemin optimal peut se reconstruire en utilisant le pointeur:

$$p(x_n^*), \quad \text{tel que: } x_n^* = \operatorname{argmin}\{c(x_n) : x_n \in D_n\}.$$

Le prochain résultat donne la complexité de la phase 2.

Proposition 4.4.5 *La récurrence "en avant", de la phase 2, donne un chemin optimal en temps $O(\theta)$ si les ensembles D_1 et $R(x_j)$, pour $x_j \in D_j, v_j \in V \backslash \{v_1\}$, sont disponibles.*

Preuve. L'exactitude de la récurrence est basée sur le fait que le graphe est acyclique, de même que sur la proposition 4.4.3. Commençant par la source, les sommets sont traités dans l'ordre topologique.

Si $\Psi(v_1, x_0) \notin D_1$ alors la proposition 4.4.3 implique qu'aucun chemin v_1-v_n réalisable n'existe et la récurrence s'arrête. Sinon, à tout sommet $v_j \in V \backslash \{v_1\}$, un plus court chemin v_1-v_j (s'il en existe) dont le prolongement peut donner un chemin réalisable au puits (i.e., avec un vecteur de ressource x_j, au sommet v_j, qui soit dans D_j) est déterminé pour chaque vecteur de ressource x_j pouvant être obtenu d'un prédécesseur quelconque v_i de v_j.

Ces chemins sont les seuls qui seront considérés pour de futures extensions à partir de v_j. La proposition 4.4.3 implique que tous les chemins v_1-v_n réalisables, qui utilisent un arc quelconque $(v_i, v_j) \in A$, sont considérés lorsque le sommet v_j est traité. Par conséquent la récurrence donne un chemin optimal.

Le calcul de $c(x_1)$ requiert un temps $O(|\mathcal{R}|)$ puisque c'est la complexité pour calculer $\Psi(v_1, x_0)$ et pour vérifier si $x_1 \in D_1$. $O(\theta)$ opérations sont nécessaires pour

calculer $c(x_1')$, pour $x_1' \in D_1 \backslash \{x_1\}$, puisque $D_1 = U_1 = \bigcup_{(v_1,v_j)\in A} U_{1j}$ et $|U_{1j}| \leq \theta_{1j}$ pour tout arc $(v_1, v_j) \in A$ (voir la preuve de la proposition 4.4.4).

Soit maintenant un sommet $v_j \in V \backslash \{v_1\}$ et un vecteur $x_j \in D_j$, la valeur $c(x_j)$ et le pointeur $p(x_j)$ sont déterminés en examinant les vecteurs qui sont dans $R(x_j)$. Chaque vecteur $x_i \in R(x_j)$ est traité en temps constant puisqu'un lien a été établi de x_j (en v_j) à x_i (au sommet correspondant v_i, voir la preuve de la proposition 4.4.4).

La complexité globale pour trouver $c(x_j)$ et $p(x_j)$ pour tous les x_j dans le graphe est donc de $O(\theta)$, d'après la proposition 4.4.4, si les ensembles D_j sont considérés dans l'ordre croissant de leurs indices. □

Nous donnons à présent la description de l'algorithme.

Algorithme en deux phases

Phase 1

1. Calculer $\underline{\varphi}_{jr}$ et $\overline{\varphi}_{jr}$, pour $r \in \mathcal{R}$ et $v_j \in V$ par ordre croissant des indices des sommets, en utilisant les équations (4.1) et (4.2). Poser $U_j = \emptyset$ pour $v_j \in V \backslash \{v_n\}$ et $U_{ij} = \emptyset$ pour $(v_i, v_j) \in A$, ainsi que $i = n - 1$.
2. Tant que $i \geq 1$:
   (a) pour $v_j \in V$ tel que $(v_i, v_j) \in A$, introduire dans U_i et dans U_{ij}, par ordre lexicographique, chaque $x_i \in \mathbf{Z}^{|\mathcal{R}|}$ tel que $\max\{\underline{\varphi}_{ir}, \underline{w}_{ijr}\} \leq x_{ir} \leq \min\{\overline{\varphi}_{ir}, \overline{w}_{ijr}\}$, pour $r \in \mathcal{R}$, si $v_j = v_n$ ou si $\Psi(v_j, x_i + u_{ij}) \in U_j$;
   (b) faire $i \leftarrow i - 1$ et retourner à 2.
3. Poser $D_1 = U_1$ et $D_j = \emptyset$, pour $v_j \in V \backslash \{v_1\}$, ainsi que $j = 2$.
4. Tant que $j \leq n$:
   (a) pour $v_i \in V$ tel que $(v_i, v_j) \in A$ et pour chaque $x_i \in U_{ij}$ tel que $x_i \in D_i$:
      i. poser $x_j = \Psi(v_j, x_i + u_{ij})$;

ii. introduire x_j dans D_j, en utilisant l'ordre lexicographique, si une copie n'existe pas déjà;

iii. introduire x_i dans $R(x_j)$ en créant un lien de x_j (en v_j) à x_i (en v_i);

(b) faire $j \leftarrow j+1$.

Phase 2

1. Calculer $x_1 = \Psi(v_1, x_0)$. Si $x_1 \notin D_1$ FIN. Sinon, poser $c(x_1) = 0$ et $c(x_1') = +\infty$ pour $x_1' \in D_1 \backslash \{x_1\}$, ainsi que $j = 2$.

2. Tant que $j \leq n$:

   (a) pour chaque $x_j \in D_j$: parcourir $R(x_j)$ pour déterminer $v_{i^\bullet}$ tel que $x_{i^\bullet} \in \text{argmin}\{c(x_i) + c_{ij} : x_i \in R(x_j)\}$, puis poser $c(x_j) = c(x_{i^\bullet}) + c_{i^\bullet j}$ et $p(x_j) = v_{i^\bullet}$;

   (b) faire $j \leftarrow j+1$.

3. Sélectionner x_n^* et $p(x_n^*)$ tels que $x_n^* \in \text{argmin}\{c(x_n) : x_n \in D_n\}$.

4.4.4 Réoptimisation

Supposons que certains sommets sont enlevés du graphe et l'on doit, de nouveau, déterminer un chemin optimal sur le graphe résiduel. Soit V' l'ensemble des sommets restants et posons à $+\infty$, le coût sur chaque arc $(v_i, v_j) \in A$ quittant un sommet $v_i \in V \backslash V'$, i.e., un sommet ne faisant plus partie du graphe. La procédure de la phase 2 ci-dessus donne toujours un chemin optimal (en moins de temps que sur le graphe initial) si V est remplacé par V' dans la récurrence.

D'une manière similaire, la procédure de la phase 2 permet de trouver un chemin optimal (s'il en existe) qui soit contraint de passer par un arc donné si le coût sur cet arc est négatif mais de valeur absolue M suffisamment grande. Par exemple M, peut être égal à $2n$ fois la plus grande valeur absolue des coûts sur les arc. Il

est clair que s'il n'existe pas de chemin réalisable passant par l'arc spécifié, alors le chemin trouvé ne contiendra pas cet arc. Ceci peut être vérifié en $O(n)$ opérations après avoir effectué la récurrence de la phase 2.

Le résultat suivant est donc immédiat, à partir des proposition 4.4.4 et 4.4.5.

Théorème 4.3 *Une première résolution de RCSPP requiert $O(\theta\,|\mathcal{R}|)$ opérations. Toute résolution subséquente nécessite un temps $O(\theta)$, si le vecteur de ressource initial x_0 ou les coûts sur les arcs sont modifiés ou si certains sommets ou arcs sont interdits ou fixés pour le chemin optimal.*

Rappelons que le paramètre θ, dans la complexité des phases 1 et 2, est inférieur ou égal au paramètre φ dans la complexité de l'algorithme de programmation dynamique. En outre, puisque la proposition 4.4.3 garantie que seuls les sous-chemins pouvant être prolongés en chemins v_1-v_n réalisables sont considérés durant la phase 2, la charge de calcul de la phase 2 sera, en pratique, nettement plus faible que celle de la phase 1 (voir la section 4.5).

Une fois la phase 1 effectuée, seule la phase 2 est nécessaire en cas de réoptimisation. Ceci confère un avantage potentiel à l'approche en deux phases par rapport à la programmation dynamique si le problème de plus court chemin avec fenêtres de ressource doit être résolu de manière répétitive.

Par ailleurs, en examinant le comportement de pire cas de l'algorithme en deux phases, on constate que la complexité de l'algorithme d'étiquetage permanent présenté dans Desrochers et Soumis [20] et dans Desrosiers *et al.* [22] est sur-évaluée (voir la section 1.3.2).

En effet, les auteurs donnent une complexité de $O(D^2)$ où $D = \sum_{v_i \in V}(b_{i1} - a_{i1} + 1)$ pour le problème de plus court chemin avec fenêtres de temps, alors que, d'après la section 4.1 (et la section 4.4.1), nous avons $\theta \leq (n-1)D < D^2$. Puisque l'algorithme d'étiquetage permanent ne retient que les vecteurs de ressources non dominés à chaque sommet, le nombre d'étiquettes à considérer est nécessairement

borné par θ. La différence θ et D^2 est particulièrement importante quand les fenêtres de ressource sont relativement large, ou quand les mises à jour des consommations de ressource ne sont pas permises, puisque, dans ce cas, la formulation discutée dans Desrochers et Soumis [20] et dans Desrosiers *et al.* [22] implique un dédoublement de ressource.

4.4.5 Exemple

Nous présentons ci-après une illustration des phases 1 et 2 de l'agorithme en utilisant le graphe de la figure 4.4.

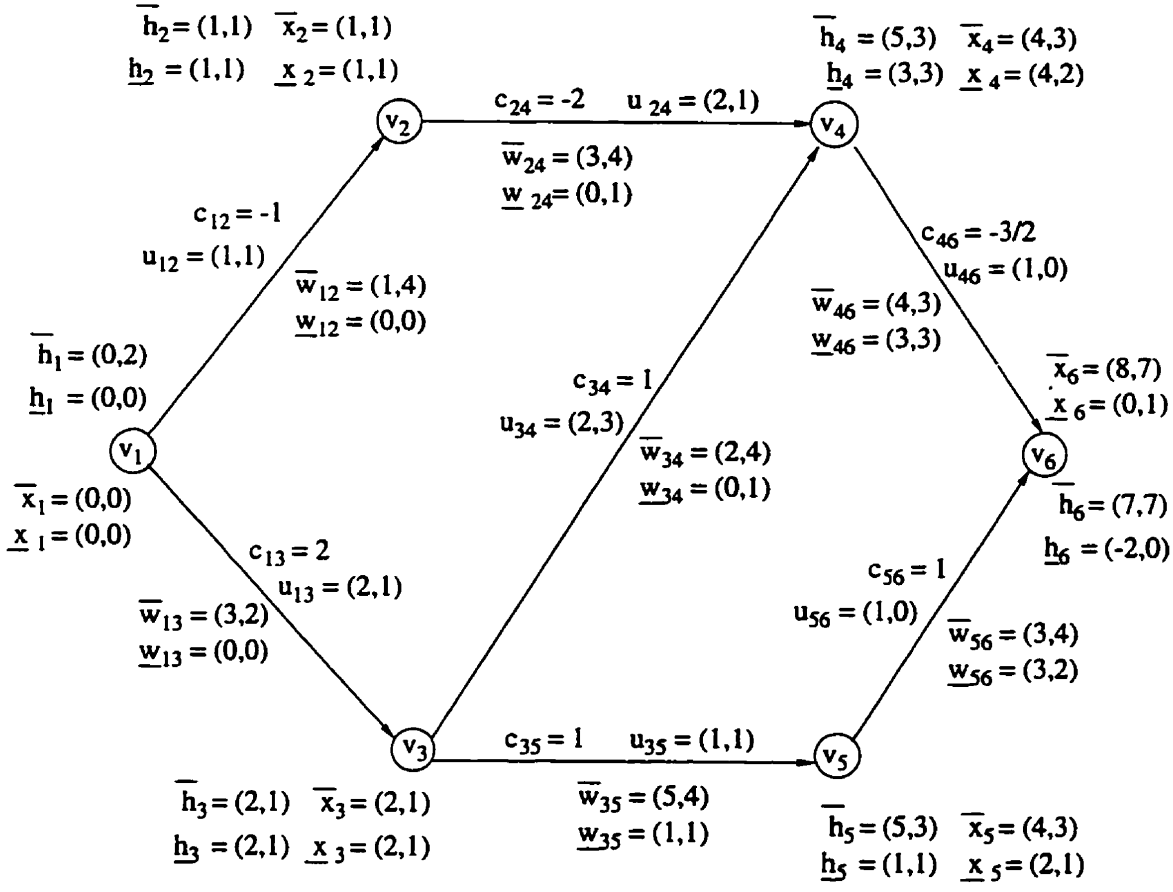


Figure 4.4 – *Illustration de l'algorithme en deux phases*

Phase 1

Étape (1):

	v_1	v_2	v_3	v_4	v_5	v_6
$(\overline{\varphi}_{j1}, \overline{\varphi}_{j2})$	$(0,2)$	$(1,1)$	$(2,1)$	$(4,3)$	$(3,2)$	$(8,7)$
$(\underline{\varphi}_{j1}, \underline{\varphi}_{j2})$	$(0,0)$	$(1,1)$	$(2,1)$	$(3,2)$	$(3,2)$	$(0,1)$

$U_1 = U_2 = U_3 = U_4 = U_5 = \emptyset,$
$U_{12} = U_{13} = U_{24} = U_{34} = U_{35} = U_{46} = U_{56} = \emptyset, \quad i = 5.$

Étape (2):

Itération	i	U_i	U_{ij}, for $(v_i, v_j) \in A$
1	5	$\{(3,2)\}$	$U_{56} = \{(3,2)\}$
2	4	$\{(3,3),(4,3)\}$	$U_{46} = \{(3,3),(4,3)\}$
3	3	$\{(2,1)\}$	$U_{35} = \{(2,1)\}, \quad U_{34} = \{(2,1)\}$
4	2	$\emptyset$	$U_{24} = \emptyset$
5	1	$\{(0,0),(0,1),(0,2)\}$	$U_{12} = \emptyset, \quad U_{13} = \{(0,0),(0,1),(0,2)\}$

Étape (3):

$$D_1 = \{(0,0),(0,1),(0,2)\},\ D_2 = D_3 = D_4 = D_5 = D_6 = \emptyset, \quad j = 2.$$

Étape (4):

Itér.	j	D_j	$R(x_j)$, for $x_j \in D_j$
1	2	$\emptyset$	–
2	3	$\{(2,1)\}$	$R(2,1) = \{((0,0),v_1),((0,1),v_1),((0,2),v_1)\}$
3	4	$\{(4,3)\}$	$R(4,3) = \{((2,1),v_3)\}$
4	5	$\{(3,2)\}$	$R(3,2) = \{((2,1),v_3)\}$
5	6	$\{(4,2),(5,3)\}$	$R(4,2) = \{((3,2),v_5)\}, \quad R(5,3) = \{((4,3),v_4)\}.$

Phase 2

Initialisation (Étape (1)):

$x_0 = (0,0), \quad x_1 = \Psi(v_1,(0,0)) = (0,0) \in D_1.$
$c(x_1) = 0,\ c(0,1) = +\infty,\ c(0,2) = +\infty; \quad j = 2.$

Étape (2):

Itération	j	$c(x_j)$	$p(x_j)$	$c(x_j)$	$p(x_j)$
1	2	–	–		
2	3	$c(2,1) = 2$	$p(2,1) = v_1$	–	–
3	4	$c(4,3) = 3$	$p(4,3) = v_3$	–	–
4	5	$c(3,2) = 3$	$p(3,2) = v_3$	–	–
5	6	$c(4,2) = 4$	$p(4,2) = v_5$	$c(5,3) = 3/2$	$p(5,3) = v_4$

Étape (3): chemin optimal: $P_{opt} = \{(v_1,v_3);(v_3,v_4);(v_4,v_6)\}, \quad$ coût $= 3/2$.

Dans cet exemple, seuls les chemins, dont le vecteur de ressource initial mis à jour ($\Psi(v_1,x_0)$) a une valeur de 0 pour la première ressource et de 0, 1 ou 2 pour la seconde, peuvent donner des chemins réalisables au puits. En plus, un tel chemin ne peut visiter le sommet v_2, puisque $D_2 = \emptyset$. On peut également remarquer que le chemin $\{(v_1,v_3);(v_3,v_5);(v_5,v_6)\}$ est aussi réalisable alors que P_{opt} ne sera plus réalisable si les contraintes de mise à jour ne sont pas actives au sommet v_4.

De telles contraintes de mise à jour sont d'une grande importance dans certaines applications des plus courts chemins avec contraintes de ressource. C'est en particulier le cas en confection d'horaires de personnel où certaines variables du problème doivent être mises à jour après des jours de congé (voir le chapitre 5). Nous discutons dans la prochaine section quelques variantes de la structure des fenêtres de ressource, susceptibles d'être rencontrées en pratique.

4.4.6 Quelques extensions

Fenêtres de ressource multiples

Considérons une situation où, pour une ressource r et un arc (v_i, v_j) donnés, il y a plusieurs intervalles de réalisabilité $[\underline{w}_{ijr},\ \underline{w}_{ijr}]$ avec la condition que les contraintes de fenêtres de ressource (1.3) doivent être satisfaites pour au moins l'un d'entre eux, sur tout chemin réalisable passant par (v_i, v_j). Ceci est illustré, dans l'application aux horaires d'infirmières, discutée au chapitre 5, par la ressource utilisée pour traiter les rotations entre les affectations de jour, du soir et de nuit.

Les algorithmes présentés dans ce chapitre s'appliquent encore, moyennant quelques modifications mineures. Soit W_{ijr} l'ensemble de toutes les paires $(\underline{w}_{ijr}, \overline{w}_{ijr})$ associées à l'arc (v_i, v_j), pour une ressource r donnée. La plus petite valeur de $\underline{w}_{ijr}$ et la plus grande valeur de $\overline{w}_{ijr}$ sur l'arc (v_i, v_j) doivent être considérées, respectivement, comme $\underline{w}_{ijr}$ et $\overline{w}_{ijr}$ lors du calcul de $\underline{\varphi}_{ijr}$ et de $\overline{\varphi}_{ijr}$. Les conditions $(\underline{w}_{ijr}, \overline{w}_{ijr}) \in W_{ijr},\ r \in \mathcal{R}$, doivent aussi être introduites dans les différentes récurrences, chaque fois que $\underline{w}_{ijr} \leq x_{ir} \leq \overline{w}_{ijr}$ est impliquée, i.e., le calcul doit être répété pour chaque fenêtre défini sur l'arc.

Mise à jour dépendant des arcs

Une autre extension du problème consiste à associer les valeurs de seuils et de mise à jour des ressources aux arcs plutôt qu'aux sommets. Cela revient à définir $\underline{h}_{ijr}$, $\overline{h}_{ijr}$, $\underline{x}_{ijr}$, et $\overline{x}_{ijr}$ pour chaque arc $(v_i, v_j) \in A$ et chaque ressource $r \in \mathcal{R}$. Ces valeurs remplacent $\underline{h}_{jr}$, $\overline{h}_{jr}$, $\underline{x}_{jr}$, et $\overline{x}_{jr}$ dans le calcul de $x_j = \Psi(v_j, x_i + u_{ij})$, si un chemin v_1-v_i de vecteur de consommation de ressource x_i est prolongé en utilisant l'arc $(v_i, v_j) \in A$.

Les seules modifications aux récurrences "en avant", discutées précédemment, consistent à remplacer $\underline{\varphi}_{1r} = \min\{\underline{h}_{1r}, \underline{x}_{1r}, \overline{x}_{1r}\}$, $\overline{\varphi}_{1r} = \max\{\overline{h}_{1r}, \underline{x}_{1r}, \overline{x}_{1r}\}$ et $x_1 = \Psi(v_1, x_0)$ par $\underline{\varphi}_{1r} = \min\{\underline{w}_{1jr} : (v_1, v_j) \in A\}$, $\overline{\varphi}_{1r} = \max\{\overline{w}_{1jr} : (v_1, v_j) \in A\}$ et $x_1 =$

x_0 respectivement, et à éliminer le test $\Psi(v_1, x_0) \in U_1$, là où c'était nécessaire. La récurrence "en arrière" de la section 4.4.2 reste inchangée.

Coûts dépendant des consommations de resources

Considérons le cas où le coût c_{ij} sur l'arc (v_i, v_j) est une fonction du vecteur de ressource x_i au sommet v_i. Par exemple, supposons que si l'arc (v_i, v_j) est utilisé, alors la mise à jour en v_i implique un coût supplémentaire. Un autre exemple consiste à appliquer une pénalité si la consommation de ressource résultant au sommet v_j s'écarte d'une valeur-cible, avant ou après mise à jour. Une discussion sur ce type de problèmes peut être trouvée dans Ioachim *et al.* [40] et dans Desaulniers *et al.* [18].

Les phases 1 et 2 restent toujours valides. Il est clair que la phase 1 ne requiert aucune modification, puisqu'aucun calcul de coût n'y est fait. La phase 2 reste également inchangée puisque tous les prédécesseurs du sommet v_j sont traités avant v_j et le calcul de $x_{i^*} = \text{argmin}\{c(x_i) + c_{ij} : x_i \in R(x_j)\}$ est effectué en examinant séquentiellement les arcs. Connaissant x_j, il suffit de calculer le coût approprié pour chaque $x_i \in R(x_j)$ individuellement et d'en retenir le minimum.

En plus, l'algorithme reste valide si la fonction de coût $c(x_i) + c_{ij}$ est remplacée par n'importe quelle fonction $f_{ij}(x_i, c(x_i))$ non décroissante par rapport à $c(x_i)$. La complexité de la phase 2 augmentera cependant d'un facteur égal à la complexité de l'évaluation de f_{ij}. Nous présentons dans la section suivantes quelques résultats numériques sur le gain en temps de calcul que permet l'algorithme en deux phases.

4.5 Tests numériques

Considérons, pour fins d'illustration, le problème de la génération d'un horaire individuel pour une personne. La personne peut effectuer chaque jour un type de quart de travail, parmi plusieurs, à moins qu'elle ne soit en congé. Les quarts diffèrent par leurs durées et les périodes couvertes (jour, soir, nuit).

Il s'agit de trouver une séquence de quarts de travail et de congés, qui minimise le coût salarial tout en satisfaisant différentes contraintes de la convention collective. Les contraintes considérées concernent la charge de travail, les congés statutaires ou de fin de semaines, la rotation entre les quarts de jour, du soir ou de nuit (en termes du nombre d'affectations consécutives de chaque type) et le pourcentage global des quarts de jour, du soir ou de nuit. Ce problème peut se formuler comme celui de plus court chemin discuté dans ce chapitre, où les sommets correspondent aux quarts de travail et les ressources sont définies de manière à tenir compte des contraintes de la conventions collective. Les détails de la modélisation seront discutés au chapitre 5.

L'algorithme a été testé sur des données en provenance du Centre des Naissances de l'Hôpital Royal Victoria de Montréal. Jusqu'à sept ressources sont considérées dans les tests. Les trois premières ressources permettent de contrôler respectivement la charge de travail, les fins de semaines et les affectations consécutives en cas de rotation. Les trois ressources suivantes servent à suivre les pourcentages de quarts de chaque type et la dernière ressource est utilisée lorsqu'un certain nombre de jours, dans une liste de candidats connus, doivent être accordés comme congés.

Tableau 4.1 – *Fenêtres d'étendues minimales et maximales*

Ressources	Seuils de mise à jour		Val. de mise à jour		Bornes de réalisab.	
	Min	Max	Min	Max	Min	Max
Charges de travail	[0, 0]	[2, 20]	[0, 0]	[2, 20]	[0, 0]	[2, 18]
Fins de semaines	[1, 1]	[-1, 1]	[-1, -1]	[-1, -1]	[1, 1]	[-1, 1]
Rotation	[1, 3]	[0, 9]	[1, 3]	[0, 9]	[0, 0]	[0, 9]
Ratio de jours	[0, 0]	[0, 28]	[0, 0]	[0, 28]	[0, 0]	[0, 6]
Ratio de nuits	[0, 0]	[0, 28]	[0, 0]	[0, 28]	[0, 0]	[0, 11]
Ratio de soirs-nuits	[0, 0]	[0, 28]	[0, 0]	[0, 28]	[0, 0]	[0, 15]
Congés statut.	[0, 0]	[0, 1]	[0, 0]	[0, 1]	[0, 0]	[0, 1]

Le tableau 4.1 donne les fenêtres de plus petite et de plus grande étendues pour chaque ressource. L'horizon considéré est de quatre semaines (28 jours) et la personne peut recevoir 5 quarts différents. Certains jours de congé sont connus *a priori* et ne

sont pas considérés dans le graphe. Les nombres de sommets et d'arcs sont donnés au tableau 4.2. Notons qu'il s'agit d'un multigraphe (certains sommets sont directement connectés par plus d'un arc), ce qui explique le nombre relativement élevé d'arcs.

Tableau 4.2 – *Performances des phases 1 et 2 en fonction du nombre de ressources*

Réseau		Ressources							Temps CPU (sec.)		Vecteurs de ressource		
Nb. de sommets	Nb. d'arcs	1	2	3	4	5	6	7	Phase1	Phase2	Total	Nb. de restants.	% de restants.
134	8808	+	+	-	-	-	-	-	0.460	0.010	52862	768	1.453
134	8808	+	+	+	-	-	-	-	1.360	0.010	145251	1240	0.854
134	8808	+	+	+	+	+	-	-	104.690	0.180	12252159	18252	0.149
134	8808	+	+	+	+	-	+	-	134.880	0.260	15674263	24215	0.154
134	8808	+	+	+	+	-	+	+	76.950	0.090	7812785	9617	0.123

Tableau 4.3 – *Performances des phases 1 et 2 en fonction de la taille du réseau*

Réseau		Ressources							Temps CPU (sec.)		Vecteurs de ressource		
Nb. de sommets	Nb. d'arcs	1	2	3	4	5	6	7	Phase1	Phase2	Total	Nb. de restants.	% de restants.
134	8808	+	+	+	-	-	-	-	1.360	0.010	145251	1240	0.854
274	21592	+	+	+	-	-	-	-	3.490	0.020	371992	2791	0.750
554	47240	+	+	+	-	-	-	-	7.810	0.060	826074	5893	0.713
1114	98536	+	+	+	-	-	-	-	16.440	0.130	1734238	12097	0.698
2234	201128	+	+	+	-	-	-	-	33.970	0.250	3550566	24505	0.690

Les tests ont été effectués sur une station de travail Sun Ultra 2, en variant, de 2 à 6, le nombre de ressources simultanément actives. Cela correspond aux situations rencontrées en pratique. Les tableaux 4.2 et 4.3 donnent les temps de calcul pour les phases 1 et 2. On y trouve également le nombre total de vecteurs de ressource examinés, de même que le nombre et le pourcentage des vecteurs restant après la phase 1. Les ressources, de 1 à 7, correspondent respectivement à la charge de travail,

aux fins de semaines, aux rotations, aux ratios de quarts de jour, de nuit et de soir-nuit, ainsi qu'aux congés statutaires. Un signe + indique une ressource active et un − le contraire. Le tableau 4.2 donne les résultats pour un horizon d'un mois en considérant les sept ressources. Dans le tableau 4.3, seules les trois premières ressources sont prises en compte, pour un horizon dont la longueur est le double de celle de la ligne précédente.

Ces résultats suggèrent que, sur les problèmes relativement difficiles, moins de 1% des vecteurs de ressource sont associés à des chemins réalisables. Cela correspond à un gain considérable de temps de calcul en cas de réoptimisation. L'efficacité de la méthode élimine pratiquement la nécessité d'une heuristique lorsque ce problème est incorporé dans un processus de génération de colonnes. Ce cas peut notamment se présenter lorsque les horaires générés doivent satisfaire les contraintes de demandes impliquant plusieurs employés simultanément. Le chapitre suivant est consacré à la modélisation et à la résolution pratique de ce problème plus général.

CHAPITRE 5

APPLICATION AUX HORAIRES DE PERSONNEL SOIGNANT

Nous montrons dans ce chapitre que la formulation du problème de plus court chemin avec fenêtres de ressource discutée au chapitre précédent permet de modéliser aisément le problème d'horaires de personnel soignant décrit à la section 1.4. La section 5.1, ci-après, est consacrée au problème de la confection d'horaires réalisables pour une personne. Un modèle est présenté à la section 5.2 pour regrouper les horaires individuels afin d'obtenir une configuration réalisable par rapport aux quotas de présences. Le reste du chapitre traite d'une procédure d'énumération implicite qui permet d'améliorer d'une manière itérative la configuration obtenue.

5.1 Génération d'horaires individuels

La réalisabilité d'un horaire individuel est définie à partir des contraintes de la convention collective s'appliquant à la personne concernée. Les contraintes considérées correspondent aux principales règles utilisées par les infirmières-chefs à l'hôpital Royal Victoria de Montréal lors de la confection des horaires.

Il s'agit des règles relatives à la charge de travail, aux congés de fins de semaines, à la rotation entre les affectations de jour, du soir et de nuit, aux pourcentages respectifs de ces différents types d'affectations et enfin aux congés statutaires. Le modèle est formulé de sorte qu'un horaire individuel corresponde à un chemin réalisable de la source au puits, dans un graphe ayant la structure décrite à la section 1.3.

5.1.1 Sommets et arcs

Soit un graphe orienté acyclique, $G = (V, A)$, avec $n = |V|$ sommets $v_1, v_2, ..., v_n$ et $m = |A|$ arcs. Un sommet v_i correspond à une affectation au quart de travail t_i le jour d_i. Les sommets sont supposés ordonnés dans le temps, avec v_1 et v_n représentant la source et le puits respectivement.

Ainsi, t_1 et d_1 se réfèrent à la dernière affectation reçue par l'infirmière au cours de l'horizon précédent, alors que t_n et d_n sont des affectations fictives. Les sommets correspondant à des congés déjà accordés ou à des quarts de travail inacceptables pour la personne considérée, sont éliminés du graphe.

Un arc (v_i, v_j) est dans A si, selon les règles de la convention collective, l'infirmière peut recevoir le quart t_i le jour d_i puis le quart t_j le jour d_j, tout en ayant un repos ou un congé durant tout le temps compris entre ces deux affectations. Notons que si $d_j = d_i + 1$ alors l'infirmière n'a pas de congé, i.e., les affectations sont consécutives, sinon, elle est en congé durant les jours $d_i + 1$ à $d_j - 1$.

Le nombre de jours de congé dans ce dernier cas doit, cependant, être égal ou supérieur au nombre minimum de jours de congé que la personne peut recevoir après une suite d'affectations consécutives se terminant par un quart t_i. Cette borne peut dépendre des quarts de travail t_i et t_j, ainsi que des jours d_i et d_j.

Nous nous référerons au problème de plus court chemin avec fenêtres de ressource RCSPP, introduit à la section 1.3 pour caractériser les séquences d'affectations admissibles, en accord avec les règles de la convention collective considérées. Les ressources de l'ensemble $\mathcal{R}$, associé au graphe $G = (V, A)$, permettent de contrôler adéquatement les caractéristiques de l'horaire comme, par exemple, la charge de travail, les fins de semaines travaillées et les rotations.

Les seuils et les valeurs de mise à jour sont utilisés pour réajuster, entre autres, les compteurs d'affectations consécutives de même type, par exemple après un congé. Le vecteur de ressource initial x_0 permet d'initialiser le chemin avec de l'information relative à l'horaire reçu par l'infirmière au cours de l'horizon précédent.

Le coût et le vecteur de ressource associés à un sous-chemin donné, caractérisent la séquence d'affectations reçues jusque là. Les fenêtres de ressource, sur un arc (v_i, v_j) quelconque, sont satisfaites uniquement si l'affectation associée à v_j peut s'ajouter à celles reçues jusqu'en v_i.

Dans le but de contrôler convenablement la séquence des congés de fins de semaines, les arcs arrivant à un sommet représentant un samedi ou un dimanche en provenance de la source ou d'un sommet qui ne correspond pas à un jour de fin de semaine sont dédoublés. C'est également le cas pour les arcs permettant de sauter au moins une fin de semaine complète.

La première occurrence de chacun de ces arcs dédoublés est utilisée pour les fins de semaines consécutives où la personne travaille, tandis que l'autre copie correspond à des congés consécutifs de fins de semaines complètes. Un exemple est décrit à la section suivante en même temps que la ressource R_2 utilisée pour contrôler les fins de semaines consécutives.

Les valeurs de ressource sur les arcs quittant la source dépendent, entre autres, du quart de travail que l'infirmière a reçu le dernier jour qu'elle a travaillé au cours de l'horizon précédent. Ainsi, plus d'un arc (v_1, v_j) pourraient être nécessaires pour un sommet v_j donné, afin d'assurer une jonction correcte entre deux horizons consécutifs. C'est en particulier le cas si d_j est le premier jour de l'horizon courant et des rotations entre affectations de jour, du soir et de nuit sont quelques fois permises sans congé.

Un dédoublement d'arc est alors nécessaire. La première occurrence d'un tel arc est utilisée lorsque les quarts de travail associés aux deux bouts de l'arc sont de même type, alors que la deuxième copie correspond au cas où les types de quarts sont différents. Un exemple est présenté à la section suivante en même temps que la ressource R_3 permettant de contrôler les rotations.

5.1.2 Définition des ressources

Nous exposons ci-après les détails de la modélisation des principales règles de la convention collective qui sont considérées. Les ressources utilisées correspondent respectivement au nombre d'heures de travail cumulées, au nombre de fins de semaines consécutives, au nombre de quarts de travail consécutifs de même type, au nombre total de quarts de travail de même type et au nombre de jours de congés statutaires dans l'horaire.

En un sommet v_i, les seuils de mise à jour, $\underline{h}_{ir}$ et $\overline{h}_{ir}$, définissent l'intervalle $[\underline{h}_{ir}, \overline{h}_{ir}]$ des valeurs de la ressource r pour lequelles une (ré-)initialisation n'est pas nécessaire. Les valeurs de mise à jour, $\underline{x}_{ir}$ et $\overline{x}_{ir}$, sont utilisées pour (ré-)initialiser la ressource r si v_i est choisi mais que la valeur courante de r n'est pas dans $[\underline{h}_{ir}, \overline{h}_{ir}]$. Notons que les valeurs de $\underline{h}_{ir}$ et $\overline{h}_{ir}$ sont respectivement égales à celles de $\underline{x}_{ir}$ et $\overline{x}_{ir}$ pour la charge de travail, les rotations et les congés statutaires, mais pas toujours pour les fins de semaines et les pourcentages de quarts de jour, du soir ou de nuit.

Lorsque les affectations correspondant à deux sommets v_i et v_j sont chosies sans congé, la fenêtre $[\underline{w}_{ijr}, \overline{w}_{ijr}]$, sur l'arc (v_i, v_j), définit les valeurs de la ressource r qui sont acceptables au sommet v_i, après (ré-)initialisation. La valeur de r varie alors de la quantité u_{ijr} avant une éventuelle ré-initialisation en v_i.

Ressource R_1: charge de travail

Une spécification typique de la charge de travail est que la personne doit travailler au moins $\underline{R}_1^i$ unités de temps (par exemple, des heures) et au plus $\overline{R}_1^i$ unités pour le bloc de deux semaines (prédéfini) contenant un sommet v_i donné. En pratique, $\underline{R}_1^i = \overline{R}_1^i$ pour le personnel permanent, s'il n'y a pas de vacance ou de congé statutaire. La ressource R_1 est utilisée pour compter le nombre total d'heures cumulées durant chaque bloc de deux semaines. La durée du quart t_i associé à un sommet v_i sera notée $R_1(t_i)$, avec $R_1(t_i) = 0$ si v_i est la source ou le puits.

Si le sommet v_i correspond à la source, au puits ou au dernier jour d'un bloc de deux semaines, on a $[\underline{h}_{i1},\ \overline{h}_{i1}] = [0,\ 0]$ et $\underline{x}_{i1} = \overline{x}_{i1} = 0$. Sinon, $[\underline{h}_{i1},\ \overline{h}_{i1}] = [R_1(t_i),\ \overline{R}_1^i]$ et $\underline{x}_{i1} = R_1(t_i),\ \overline{x}_{i1} = \overline{R}_1^i$.

Soit un arc (v_i, v_j). Nous présentons d'abord le cas où v_i correspond à la source ou au dernier jour d'un bloc de deux semaines. Si v_j est le puits alors $[\underline{w}_{ij1},\ \overline{w}_{ij1}] = [0,\ 0]$ et $u_{ij1} = 0$. Sinon, supposons que v_j correspond au dernier jour d'un bloc de deux semaines. On a alors $u_{ij1} = -\overline{R}_1^j$. Si v_j est dans le même bloc de deux semaines que v_i, on a $[\underline{w}_{ij1},\ \overline{w}_{ij1}] = [\underline{R}_1^j - R_1(t_j),\ \overline{R}_1^j - R_1(t_j)]$, sinon $\underline{w}_{ij1} = 0$ ou $\underline{w}_{ij1} = +\infty$ selon que $\underline{R}_1^j \leq R_1(t_j)$ ou non, et $\overline{w}_{ij1} = 0$ ou $\overline{w}_{ij1} = -\infty$ selon que $\overline{R}_1^j \geq R_1(t_j)$ ou non. Supposons que v_j ne correspond ni au puits ni au dernier jour d'un bloc de deux semaines. On a alors $u_{ij1} = R_1(t_j)$. Si v_j est dans le même bloc de deux semaines que v_i, on a $[\underline{w}_{ij1},\ \overline{w}_{ij1}] = [R_1(t_i),\ \overline{R}_1^j - R_1(t_j)]$, sinon $[\underline{w}_{ij1},\ \overline{w}_{ij1}] = [0,\ 0]$.

Considérons maintenant le cas où v_i n'est ni la source ni le dernier jour d'un bloc de deux semaines. Si v_j est le puits alors $[\underline{w}_{ij1},\ \overline{w}_{ij1}] = [\underline{R}_1^i,\ \overline{R}_1^i]$ et $u_{ij1} = 0$. Sinon, supposons que v_j est le dernier jour d'un bloc de deux semaines. On a alors $u_{ij1} = -\overline{R}_1^j$. Si v_j est dans le même bloc de deux semaines que v_i, alors $[\underline{w}_{ij1},\ \overline{w}_{ij1}] = [\underline{R}_1^j - R_1(t_j),\ \overline{R}_1^j - R_1(t_j)]$, sinon $\underline{w}_{ij1} = \underline{R}_1^i$ ou $\underline{w}_{ij1} = +\infty$ selon que $\underline{R}_1^j \leq R_1(t_j)$ ou non, et $\overline{w}_{ij1} = \overline{R}_1^i$ ou $\overline{w}_{ij1} = -\infty$ selon que $\overline{R}_1^j \geq R_1(t_j)$ ou non. Supposons que v_j n'est ni le puits ni le dernier jour d'un bloc de deux semaines. On a alors $u_{ij1} = -\overline{R}_1^i - 1$. Si v_j est dans le même bloc de deux semaines que v_i, on a $[\underline{w}_{ij1},\ \overline{w}_{ij1}] = [R_1(t_i),\ \overline{R}_1^j - R_1(t_j)]$, sinon $[\underline{w}_{ij1},\ \overline{w}_{ij1}] = [\underline{R}_1^i,\ \overline{R}_1^i]$.

Ressource R_2: congés de fins de semaines

Cette ressource est utilisée pour compter les fins de semaines consécutives, de manière à s'assurer que l'infirmière travaille un certain nombre de fins de semaines consécutives, puis est en congé durant un nombre donné de fins de semaines consécutives. Les valeurs positives de R_2 correspondent aux fins de semaines où l'infirmière travaille et les valeurs négatives aux congés. Les bornes sur le nombre de fins de semaines consécutives où la personne peut travailler sont $\underline{R}_2$ et $\overline{R}_2$. Les valeurs correspondantes pour les congés de fins de semaines sont $\underline{R}_2'$ et $\overline{R}_2'$.

Il est nécessaire d'effectuer des mises à jour lors d'une transition d'une suite de congés de fins de semaines consécutives à une suite de fins de semaines où la personne travaille, ou vice versa. Pour cela, les arcs arrivant à un sommet de fin de semaine, en provenance de la source ou d'un sommet qui ne correspond pas à un jour de fin de semaine, sont dédoublés. Il en est de même pour les arcs qui permettent de sauter une fin de semaine entière. Des consommations de ressource différentes sont associées à chaque copie d'un arc dédoublé.

Par exemple, considérons une infirmière qui a un congé durant une fin de semaine complète donnée. La ressource R_2 doit varier de -1 si elle était également en congé la fin de semaine précédente. Sinon, la valeur de la de ressource R_2 est strictement positive avant le congé et sa variation doit être telle qu'une remise à jour à -1 soit requise après après avoir choisie l'arc liant les deux sommets impliqués. Ainsi, deux copies de l'arc doivent être disponibles, pour permettre chacune des deux variations.

Soit un sommet v_i donné. Si v_i correspond à un jour de fin de semaine, on a $[\underline{h}_{i2},\ \overline{h}_{i2}] = [1,\ \overline{R}_2]$ et $\underline{x}_{i2} = \overline{x}_{i2} = 1$. Sinon $[\underline{h}_{i2},\ \overline{h}_{i2}] = [-\overline{R}'_2,\ \overline{R}_2]$ et $\underline{x}_{i2} = \overline{x}_{i2} = -1$. Notons que la valeur de cette ressource au puits est reportée à la source pour l'horizon suivant, afin d'assurer une continuité dans l'attribution des congés de fins de semaines.

Considérons un arc dédoublé (v_i, v_j). Si v_j est un sommet de fins de semaine on a $[\underline{w}_{ij2},\ \overline{w}_{ij2}] = [0,\ \overline{R}_2 - 1]$ et $u_{ij2} = 1$ sur la premières occurrence de (v_i, v_j) et $[\underline{w}_{ij2},\ \overline{w}_{ij2}] = [-\overline{R}'_2,\ -\underline{R}'_2]$ et $u_{ij2} = 0$ sur la seconde copie. Sinon, v_j implique un congé de fin de semaine entière. On a alors $[\underline{w}_{ij2},\ \overline{w}_{ij2}] = [\underline{R}_2,\ \overline{R}_2]$ et $u_{ij2} = \overline{R}_2 + 1$ pour la premières occurrence de (v_i, v_j) et $[\underline{w}_{ij2},\ \overline{w}_{ij2}] = [-\overline{R}'_2 + 1,\ 0]$ et $u_{ij2} = -1$ pour la seconde copie.

Soit un arc non dédoublé (v_i, v_j). Si aucun des deux sommets ne correspond à un jour de fin de semaine, on a $[\underline{w}_{ij2},\ \overline{w}_{ij2}] = [-\overline{R}'_2,\ \overline{R}_2]$ et $u_{ij2} = 0$. Si v_i et v_j correspondent à des fins de semaines différentes, $[\underline{w}_{ij2},\ \overline{w}_{ij2}] = [1,\ \overline{R}_2 - 1]$ et $u_{ij2} = 1$. Pour tous les autres cas, on a $[\underline{w}_{ij2},\ \overline{w}_{ij2}] = [1,\ \overline{R}_2]$ et $u_{ij2} = 0$.

Ressource R_3: rotations

La rotation consiste à se déplacer entre les quarts de jour, du soir et de nuit. La rotation peut, dans le cas général, être permise avec ou sans congé. Le nombre d'affectations consécutives que l'infirmière doit recevoir avant une rotation est généralement limité par une borne inférieure et une borne supérieure pour les différents types de quarts. Nous utiliserons $\underline{R}_3^D$ et $\overline{R}_3^D$, $\underline{R}_3^E$ et $\overline{R}_3^E$, ainsi que $\underline{R}_3^N$ et $\overline{R}_3^N$ pour représenter ces limites inférieures et supérieures pour les affectations de jour, du soir et de nuit respectivement. La ressource R_3 est utilisée pour compter le nombre d'affectations consécutives correspondant au même type de quart de travail.

Les intervalles $[0,\ \overline{R}_3^D]$, $[\overline{R}_3^D + 1,\ \overline{R}_3^D + \overline{R}_3^E]$ et $[\overline{R}_3^D + \overline{R}_3^E + 1,\ \overline{R}_3^D + \overline{R}_3^E + \overline{R}_3^N]$ correspondent aux valeurs de R_3 pour les quarts de jour, du soir et de nuit, respectivement. La ressource R_3 doit être remise à jour après un congé ou une rotation. On a $[\underline{h}_{i3},\ \overline{h}_{i3}] = [0,\ \overline{R}_3^D + \overline{R}_3^E + \overline{R}_3^N]$, $\underline{x}_{i3} = 0$ et $\overline{x}_{i3} = \overline{R}_3^D + \overline{R}_3^E + \overline{R}_3^N$ pour la source et le puits, et $[\underline{h}_{i3},\ \overline{h}_{i3}] = [1,\ \overline{R}_3^D]$, $\underline{x}_{i3} = 1$ et $\overline{x}_{i3} = \overline{R}_3^D$ pour un sommet correspondant à un quart de jour. Pour les sommets associés à des quarts du soir ou de nuit, ces valeurs sont égales aux bornes correspondantes pour la ressource R_3.

Supposons que l'infirmière reçoit un quart de jour, le premier jour de l'horizon. La ressource R_3 aura une croissance unitaire si l'infirmière avait également reçu un quart de jour, le dernier jour de l'horizon précédent. Si elle avait plutôt reçu un quart de nuit, alors la valeur de R_3 doit être posée à 1, i.e., R_3 doit décroître d'une quantité égale à sa valeur courante moins 1. Par conséquent, un dédoublement d'arcs est nécessaire entre la source et les sommets correspondant au premier jour de l'horizon, si la rotation est permise sans congé (comme mentionné à la section 5.1.1).

La fenêtre de la ressource R_3 est $[0,\ \overline{R}_3^D - 1]$ sur la première occurrence de l'arc (v_1, v_i) tel que d_i est le premier jour de l'horizon et t_i est un quart de jour; et $[\overline{R}_3^D + \underline{R}_3^E,\ \overline{R}_3^D + \overline{R}_3^E] \cup [\overline{R}_3^D + \overline{R}_3^E + \underline{R}_3^N,\ \overline{R}_3^D + \overline{R}_3^E + \overline{R}_3^N]$ pour la seconde copie. Si t_i est plutôt un quart du soir, la fenêtre est $[0,\ 0] \cup [\overline{R}_3^D + 1,\ \overline{R}_3^D + \overline{R}_3^E - 1]$ pour la première occurrence de l'arc et $[\underline{R}_3^D,\ \overline{R}_3^D] \cup [\overline{R}_3^D + \overline{R}_3^E + \underline{R}_3^N,\ \overline{R}_3^D + \overline{R}_3^E + \overline{R}_3^N]$ pour la deuxième.

Si le sommet v_i correspond à un quart de nuit du premier jour de l'horizon, la fenêtre est $[0,\ 0] \cup [\overline{R}_3^D + \overline{R}_3^E + 1,\ \overline{R}_3^D + \overline{R}_3^E + \overline{R}_3^N - 1]$ pour la première occurrence de l'arc (v_1, v_i) et $[\underline{R}_3^D,\ \overline{R}_3^D] \cup [\overline{R}_3^D + \underline{R}_3^E,\ \overline{R}_3^D + \overline{R}_3^E]$ pour la deuxième. L'union de cette dernière fenêtre et de $[0,\ 0] \cup [\overline{R}_3^D + \overline{R}_3^E + \underline{R}_3^N,\ \overline{R}_3^D + \overline{R}_3^E + \overline{R}_3^N]$ donne la fenêtre pour chacun des arcs restants qui partent de la source.

Pour de tels arcs où la fenêtre de la ressource R_3 est la réunion de plusieurs intervalles disjoints, la plus petite et la plus grande valeurs acceptables sont respectivement considérés comme $\underline{w}_{ij3}$ et $\overline{w}_{ij3}$. Si la rotation n'est pas permise sans congé, alors les secondes copie des arcs seront supprimées.

Les arcs (v_i, v_n) tels que d_i est le dernier jour dans l'horizon, ont pour fenêtre $[1,\ \overline{R}_3^D]$ si t_i est un quart de jour. Soit un arc (v_i, v_j) tel que d_i n'est pas le dernier jour dans l'horizon. Lorsque t_i est un quart de jour, la fenêtre est $[1,\ \overline{R}_3^D - 1]$ si d_i et d_j sont des jours consécutifs et t_j est aussi un quart de jour, et $[\underline{R}_3^D,\ \overline{R}_3^D]$ sinon.

Si t_i est plutôt un quart du soir, la fenêtre est $[\overline{R}_3^D + 1,\ \overline{R}_3^D + \overline{R}_3^E - 1]$ si d_i et d_j sont des jours consécutifs et t_j est aussi un quart du soir; et $[\overline{R}_3^D + \underline{R}_3^E,\ \overline{R}_3^D + \overline{R}_3^E]$ sinon. Enfin, si t_i est un quart de nuit, nous avons $[\overline{R}_3^D + \overline{R}_3^E + 1,\ \overline{R}_3^D + \overline{R}_3^E + \overline{R}_3^N - 1]$ si d_i et d_j sont des jours consécutifs et t_j est un quart de nuit, et $[\overline{R}_3^D + \overline{R}_3^E + \underline{R}_3^N,\ \overline{R}_3^D + \overline{R}_3^E + \overline{R}_3^N]$ sinon.

La consommation de la ressource R_3 sur un arc (v_i, v_j) donné, est 1 si d_i et d_j sont des jours consécutifs et t_i et t_j sont des quart de même type, par exemple si t_i et t_j sont tous les deux des quarts de jours. La consommation est égale à zéro sur les arcs (v_i, v_n), où d_i est le dernier jour de l'horizon.

Tous les autres arcs (v_i, v_j) correspondent à des congés et impliquent une remise à jour de la ressource cumulée. La consommation sur chacun de ces arcs est égale à $\underline{x}_{j3} - \overline{w}_{ij3}$, où $\underline{x}_{j3}$ est la valeur de mise à jour au sommet v_j et $\overline{w}_{ij3}$ est la borne supérieure de la fenêtre de la ressource sur (v_i, v_j), telle que donnée précédemment.

Ressources R_4, R_5 et R_6: ratios des quarts de travail

Les contraintes de la convention collective spécifient les règles pour la proportion des quarts de jour, du soir ou de nuit, dans l'ensemble des affectations reçues durant l'horizon. Les ressources R_4, R_5 et R_6 sont utilisées pour compter les affectations de jour, du soir et de nuit respectivement. Cependant, si un seul des trois ratios est requis, une seule des deux ressources restantes est utilisée pour compter les quarts de travail qui ne correspondent pas au ratio demandé.

Par exemple, si le ratio de jour est suffisant pour satisfaire la contrainte, alors R_4 servira à compter les quarts de jour et une seule des ressources R_5 et R_6 sera active et servira à compter à la fois les quarts du soir et de nuit. Nous donnons ci-après les limites, aux sommets et sur les arcs, pour la ressource R_4. Les valeurs pour les deux autres ressources sont similaires.

Soient $\underline{R}_4$ et $\overline{R}_4$ la plus petite et la plus grande valeurs acceptables du ratio des quarts de jour par rapport au nombre total d'affectations dans l'horaire retenu. Le plus petit et le plus grand nombres d'affectations que la personne peut recevoir durant l'horizon sont notés $\underline{R}_a$ et $\overline{R}_a$ respectivement. Considérons un sommet v_i donné. Si v_i est la source, on a $[\underline{h}_{i4},\ \overline{h}_{i4}] = [0,\ 0]$ et $\underline{x}_{i4} = \overline{x}_{i4} = 0$. Si v_i est le puits, on a $[\underline{h}_{i4},\ \overline{h}_{i4}] = [0,\ \overline{R}_a]$, $\underline{x}_{i4} = 0$ et $\overline{x}_{i4} = \overline{R}_a$. Dans tous les autres cas, $\underline{h}_{i4} = \underline{x}_{i4} = 1$ si v_i correspond à un quart de jour, sinon $\underline{h}_{i4} = \underline{x}_{i4} = 0$. Si v_i est différent de la source et du puits, on a $\overline{h}_{i4} = \min\{d_i, \overline{R}_4\overline{R}_a\}$ et $\overline{x}_{i4} = \overline{R}_4\overline{R}_a + 1$.

Considérons maintenant la consommation u_{ij4} et la fenêtre $[\underline{w}_{ij4},\ \overline{w}_{ij4}]$ pour un arc (v_i, v_j) donné. Si v_j correspond à un quart de jour, on a les valeurs $u_{ij4} = 1$ et $\overline{w}_{ij4} = \min\{d_i, \overline{R}_4\overline{R}_a - 1\}$, sinon $u_{ij4} = 0$ et $\overline{w}_{ij4} = \min\{d_i, \overline{R}_4\overline{R}_a\}$. Si v_i est différent du puits et de la source, on a $\underline{w}_{ij4} = 1$ lorsque v_i correspond à un quart de jour et $\underline{w}_{ij4} = 0$ sinon. Pour la source et le puits, on a respectivement, $\underline{w}_{ij4} = 0$ et $\underline{w}_{ij4} = \underline{R}_4\underline{R}_a$.

Il faut noter cependant que cette structure de la ressource R_4 permet seulement de trouver les valeurs comprises entre le nombre minimal et le nombre maximal d'affectations de jour, pour lesquelles il peut exister un ratio satisfaisant les limites

imposées. Certaines combinaisons entre les valeurs de R_4 et celles de la ressource qui compte les autres affectations peuvent donc ne pas satisfaire les limites de ratios imposées. Par exemple, un chemin dont la consommation pour cette dernière ressource est égale à la limite inférieure mais dont la valeur de R_4 est le maximum permis pourrait ne pas être réalisable.

Pour tenir compte de cette particularité, un test supplémentaire sera introduit dans la phase 1 de l'algorithme en deux phases (voir section 4.4), lors de la caractérisation des consommations de ressource (étape 2a de l'algorithme). Le test consistera à ne garder, dans les ensembles U_{in} associés aux arcs arrivant au puits, que les vecteurs de ressource x_i qui satisfont les limites de ratios imposées.

Ressource R_7: congés statutaires

Supposons qu'au moins $\underline{R}_7$ et au plus $\overline{R}_7$ jours de congés doivent être accordés à l'infirmière à partir d'une liste D de jours candidats (par exemple, le 25 décembre et/ou le 1er janvier). La ressource R_7 représente le nombre de ces jours de congés statutaires le long du chemin.

Etant donné un sommet v_i. On a $\underline{h}_{i4} = \underline{x}_{i4} = \underline{R}_7$ si le jour d_i, correspondant à v_i, est ultérieur au dernier jour admissible de la liste D. Sinon, $\underline{h}_{i4} = \underline{x}_{i4} = 0$ pour tous les autres sommets. Si d_i est antérieur au premier jour de D, on a $\overline{h}_{i4} = \overline{x}_{i4} = 0$. Sinon, L$\underline{h}_{i4} = \underline{x}_{i4} = \overline{R}_7$.

Pour tout arc $(v_i, v_j) \in A$, la consommation u_{ij7} de la ressource R_7 est égale au nombre de jour de congé entre d_i et d_j qui appartiennent à D, et $\overline{w}_{ij4} = \overline{R}_7 - u_{ij7}$. Si d_j est ultérieur au dernier jour de D, on a $\underline{w}_{ij4} = \underline{R}_7 - u_{ij7}$, sinon, $\underline{w}_{ij4} = 0$.

5.1.3 Coûts sur les arcs

Les coûts sur les arcs sont définis de sorte que le coût total d'un chemin soit une combinaison linéaire des différents facteurs contribuant à la qualité de l'horaire. Le coût d'un arc (v_i, v_j) implique en particulier les préférences de l'infirmière, sa rémunération pour une affectation au quart t_j le jour d_j, ainsi que la préférence (du décideur) de donner cette affectation à une personne de ce niveau de qualification.

Diverses pénalités peuvent être ajoutées au coût de certains arcs indésirables tels que ceux qui impliquent une rotation sans congé. Le calcul détaillé des coûts sur les arcs tient également compte de la contribution de l'horaire dans la satisfaction des quotas de demande, i.e., sa contribution à la configuration complète des horaires de tout le personnel soignant.

Le problème de la génération d'une configuration optimale d'horaires pour tout le personnel est abordé à section 5.2 et est formulé comme un programme linéaire en variables 0-1, où les colonnes de la matrice des contraintes correspondent aux horaires individuels réalisables. Ce problème sera appelé problème maître. Le coût sur chaque arc est défini de telle manière que le coût total d'un chemin corresponde au coût réduit de la colonne associée. Ainsi, la recherche d'une variable de coût réduit négatif pour un pivot de l'algorithme du simplexe, lors de la résolution du problème maître, revient à déterminer un chemin de coût réduit négatif.

Pour garder une structure de problème de plus court chemin, pour le problème auxiliaire de génération d'horaires individuels, les coefficients utilisés dans la combinaison linéaire définissant le coût sur un arc, sont négatifs pour les préférences et positifs pour les salaires. L'interaction avec les autres horaires de la configuration se traduit par la présence de multiplicateurs duaux de signe variable. Du fait de la présence de ces variables duales, le calcul détaillé des coûts sur les arcs sera présenté à la section 5.2.3, après avoir discuté la formulation du problème maître.

Notons que la présence des contraintes de ressource utilisées pour modéliser les contraintes de la convention collective rend particulièrement ardue la résolution

du problème auxiliaire d'horaires individuels. En effet, l'opérateur, $\Psi(\cdot,\cdot)$, d'accumulation des consommations de ressource n'est pas nécessairement une fonction non décroissante pour une ressource donnée, le long du chemin. En outre, les fenêtres de ressource sur les arcs définissent des bornes rigides devant être absolument satisfaites.

L'algorithme en deux phases discuté au chapitre 4 permet cependant de résoudre efficacement ce problème complexe de cheminement. Une fois la phase 1 effectuée, seule la phase 2 est nécessaire pour toutes les résolutions subséquentes du problème auxiliaire. Par ailleurs, au lieu de déterminer seulement le chemin de coût réduit minimal, on peut retenir, à la fin de la phase 2, tous les chemins réalisables de coût réduit négatif (s'il en existe) qui arrivent au puits v_n.

A notre connaissance, aucun autre algorithme spécialisé, dans la littérature, ne peut être utilisé comme une boîte noire pour ce problème. On peut, néanmoins, envisager une généralisation de l'algorithme d'étiquetage permanent présenté dans Desrosiers *et al.* [22] pour le problème de plus court chemin avec fenêtres de ressource aux sommets. Cependant, cette approche implique un dédoublement des ressources pour satisfaire les deux bornes des fenêtres. En outre, les tests de dominance utilisés dans cet algorithme ne s'appliquent pas dans le cas présent.

5.2 Génération d'horaires de groupe

Nous discutons ici un modèle de génération de colonnes en variables 0-1 pour le problème, plus général, de la recherche d'une configuration optimale d'horaires pour l'ensemble du personnel soignant d'une unité de soins. Ce modèle contient, comme problème auxiliaire, celui de la génération d'horaires réalisables pour une infirmière donnée. Le problème maître sélectionne les horaires individuels pour satisfaire les contraintes de demande tout en minimisant le coût salarial et en maximisant les préférences personnelles et l'équilibre des équipes.

5.2.1 Formulation de base

Dans la formulation de base du problème maître, on minimise, pour le personnel permanent, une somme pondérée, comprenant les rémunérations salariales, les préférences individuelles et l'équilibre entre les personnes expérimentées et celle qui le sont moins, sous les contraintes de satisfaction des quotas de demande. Des poids positifs sont associés aux salaires tandis que les préférences et l'équilibre des équipes ont des poids négatifs.

Nous supposons que le salaire et la préférence de chaque infirmière pour chaque affectation potentielle, sont connus *a priori*. Les valeurs de préférence peuvent être obtenues, en pratique, en utilisant un système de points-quotas équitablement distribués aux infirmières, comme dans Warner [62]. Les infirmières peuvent, également, indiquer des préférences globales, telles que "seulement des quarts de jours durant la semaine x", à partir desquelles des valeurs de préférence spécifiques seront déduites pour chaque affectation potentielle.

Le salaire et le niveau de préférence d'une infirmière pour un horaire donné sont ensuite calculés en additionnant (éventuellement après une normalisation), respectivement, ses rémunérations et ses préférences pour les affectations contenues dans l'horaire. La préférence de l'infirmière pour chaque affectation et pondérée par son ancienneté. Ces calculs sont effectués durant la résolution du problème auxiliaire.

Soient f_{ks} et g_{ks}, respectivement, le niveau de préférence de l'infirmière k pour l'horaire potentiel s et sa rémunération si on lui attribue cet horaire. L'ensemble des jours de l'horizon est noté $\mathcal{D}$ et T désigne le nombre de quarts de travail. Chaque horaire potentiel s peut s'exprimer comme un vecteur de $T \times |\mathcal{D}|$ composantes a_{std} telles que:

$$a_{std} = \begin{cases} 1 & \text{si l'horaire } s \text{ implique une affectation au quart } t \text{ le jour } d \\ 0 & \text{sinon.} \end{cases}$$

On définit également les coefficients suivants:

$$b_{k\ell} = \begin{cases} 1 & \text{si l'infirmière } k \text{ a le niveau de qualification } \ell \\ 0 & \text{sinon,} \end{cases}$$

$$\alpha_{tp} = \begin{cases} 1 & \text{si le quart } t \text{ couvre la période de demande } p \\ 0 & \text{sinon.} \end{cases}$$

Les coefficients α_{tp} sont utiles pour lier les horaires aux contraintes de demande, puisqu'un horaire est défini comme une séquence de quarts de travail et les quotas sont donnés par périodes de demande. Une illustration des quarts de travail par rapport aux périodes de demande, est donnée à la figure 1.4 (section 1.4). Soient également les variables de décision suivantes:

$$y_{ks} = \begin{cases} 1 & \text{si l'infirmière } k \text{ reçoit l'horaire } s \\ 0 & \text{sinon.} \end{cases}$$

Si K est le nombre d'infirmières et S le nombre total d'horaires individuels potentiels, alors le nombre d'infirmières de niveau de qualification ℓ qui ont reçu le quart t à la période p le jour d est donné par

$$\sum_{k=1}^{K} \sum_{s=1}^{S} \alpha_{tp} b_{k\ell} a_{std} y_{ks}.$$

Le nombre d'infirmières dont le niveau de qualification fait partie d'un ensemble $\mathcal{L}$ donné et qui sont affectées à un quart de travail appartenant à l'ensemble $\mathcal{T}$, à la période p le jour d, est:

$$\sum_{\ell \in \mathcal{L}} \sum_{t \in \mathcal{T}} \sum_{k=1}^{K} \sum_{s=1}^{S} \alpha_{tp} b_{k\ell} a_{std} y_{ks}.$$

La proportion de personnes expérimentées, parmi les infirmières qui sont affectées à la même période un jour donné, est un facteur important lors de la confection d'horaires de personnel soignant. Par exemple, l'infirmière-chef peut préférer affecter une infirmière-bachelière, une stagiaire et deux auxiliaires à une certaine période pour un jour donné, plutôt que deux infirmières-bachelières et deux auxiliaires, même si les deux équipes sont acceptables.

Afin de tenir compte de ce type de spécification, on définit e^{ℓ}_{pd}, la préférence d'affecter une infirmière de niveau de qualification ℓ à la période p le jour d. Si L est le nombre de niveaux de qualification, une fonction globale d'équilibre des équipes est donnée par:

$$\sum_{k=1}^{K}\sum_{s=1}^{S}\sum_{d\in\mathcal{D}}\sum_{p\in\mathcal{P}}\sum_{\ell=1}^{L}\sum_{t=1}^{T} e^{\ell}_{pd}\alpha_{tp}b_{k\ell}a_{std}y_{ks}.$$

La fonction objectif du problème maître correspond alors à la somme pondérée suivante, sur tout l'ensemble des horaires potentiels:

$$\sum_{k=1}^{K}\sum_{s=1}^{S}(g_{ks}-f_{ks}-\sum_{d\in\mathcal{D}}\sum_{p\in\mathcal{P}}\sum_{\ell=1}^{L}\sum_{t=1}^{T} e^{\ell}_{pd}\alpha_{tp}b_{k\ell}a_{std})y_{ks}. \tag{5.1}$$

Les contraintes de partitionnement,

$$\sum_{s=1}^{S} y_{ks}=1, \quad k=1,2,...,K, \tag{5.2}$$

sont nécessaires pour assurer que chaque infirmière reçoit exactement un horaire.

Plusieurs quotas de demande peuvent être spécifiés pour une période p et un jour d donnés. Par exemple, le quota d'infirmières-bachelières devant recevoir des quarts de 8 heures peut être différent de celui de stagiaires et d'auxiliaires devant effectuer des quarts de 12 heures ou de 8 heures, etc...

Ainsi, à chaque quota de demande pour une paire (p,d) peut comprendre un sous-ensemble $\mathcal{L}$ de niveaux de qualification et un sous-ensemble $\mathcal{T}$ de quarts de travail. Soit $\mathcal{Q}(p,d)$ l'ensemble de toutes les paires $(\mathcal{L},\mathcal{T})$ associées à (p,d), et $q_{\mathcal{LT}}$ le quota pour chacune de ces paires $(\mathcal{L},\mathcal{T})$. Les contraintes de demande peuvent se formuler comme suit:

$$\sum_{\ell\in\mathcal{L}}\sum_{t\in\mathcal{T}}\sum_{k=1}^{K}\sum_{s=1}^{S}\alpha_{tp}b_{k\ell}a_{std}y_{ks} \begin{pmatrix}\leq\\=\\\geq\end{pmatrix} q_{\mathcal{LT}}, \qquad (\mathcal{L},\mathcal{T})\in\mathcal{Q}(p,d), \quad p\in\mathcal{P}, d\in\mathcal{D} \tag{5.3}$$

Les signes $\leq$, $=$, and $\geq$ représentent respectivement une satisfaction maximale, exacte ou minimale de la demande, tandis que $\mathcal{P}$ et $\mathcal{D}$ correspondent respectivement à l'ensemble des périodes de demande journalières et à celui des jours de l'horizon.

Les contraintes d'intégralité suivantes:

$$y_{ks} \in \{0,1\}, \quad k = 1,2,...,K, \quad s = 1,2,...,S, \tag{5.4}$$

sont finalement requises pour assurer que chaque infirmière reçoit entièrement un horaire donné ou pas du tout.

La formulation de base du problème maître consiste donc à minimiser la fonction objectif (5.1), sous les contraintes (5.2) à(5.4). Ce modèle permet de résoudre le problème d'horaires pour le personnel permanent. En cas d'insuffisance de celui-ci, le modèle peut être légèrement modifié pour inclure le personnel non permanent.

5.2.2 Le personnel flottant

Lorsqu'il y a trop ou trop peu d'infirmières permanentes pour satisfaire les quotas, le programme mathématique correspondant au problème maître peut ne pas avoir de solution. Dans ce cas, on peut éviter la non-réalisabilité en ajoutant des variables de déficit $y'_{\mathcal{LT}}$ et en retranchant des variables de surplus $y''_{\mathcal{LT}}$, non négatives, aux membres de gauche des contraintes de demandes (5.3).

Un coût positif de très grande valeur absolue est associé à chaque variable ainsi introduite. Ces coûts ne sont cependant pas nécessairement les mêmes pour les variables $y'_{\mathcal{LT}}$ et $y''_{\mathcal{LT}}$.

Ces variables représentent respectivement le déficit et le surplus d'infirmières pour chaque niveau de qualification et de quart de travail, chaque période et chaque jour de l'horizon. En résolvant le modèle modifié, on peut donc obtenir une estimation du temps supplémentaire requis ou du besoin en personnel flottant (si les $y'_{\mathcal{LT}}$ ne sont pas tous nuls), ou bien de la réduction de charge de travail pouvant être fait tout en satisfaisant les quotas (si les $y''_{\mathcal{LT}}$ ne sont pas tous nuls).

Les priorités d'affectations sont telles que la charge de travail régulière du personnel permanent à temps plein ou à temps partiel doit être considérée en premier lieu. Ensuite, on prend en compte le temps supplémentaire pour les infirmières permanentes à temps partiel, dans la limite d'une charge à temps plein. Finalement on considère les infirmières de l'équipe flottante basée à l'unité de soin.

Par conséquent, si des variables $y'_{\mathcal{L}\mathcal{T}}$ et $y''_{\mathcal{L}\mathcal{T}}$ non nulles sont obtenues, on augmentera d'abord la charge de travail maximale des infirmières permanentes à temps partiel désireuses de faire du temps supplémentaire. Ensuite, si nécessaire, le personnel flottant basé à l'unité est introduite, en tenant compte de l'ancienneté dans les priorités.

Notons que dans une formulation matricielle du problème maître, une colonne A_{ks}, a 0 sur ses K premières lignes sauf sur la ligne k, qui contient un 1. Chaque ligne d'indice $(\mathcal{L}, \mathcal{T}, p, d)$ a une valeur égale à:

$$\sum_{\ell \in \mathcal{L}} \sum_{t \in \mathcal{T}} \alpha_{tp} b_{k\ell} a_{std}.$$

Chaque colonne A_{ks} correspond à un horaire réalisable pour une infirmière k et doit donc satisfaire les contraintes de la convention collective, telles qu'elles s'appliquent à cette infirmière. Ces contraintes additionnelles sont prises en compte dans la structure du problème auxiliaire (voir la section 5.1).

Étant donné une solution courante du problème maître, la recherche d'un nouvel horaire doit tenir compte du niveau de satisfaction actuelle des quotas. Ceci peut se faire en incorporant les multiplicateurs duaux, associés à la solution courante, dans le calcul des coûts sur les arcs dans le problème auxiliaire.

5.2.3 Calcul des coûts sur les arcs

Soient λ le vecteur de variables duales associées à la solution optimale d'une relaxation linéaire du problème maître, et W_{ks} le poids total de la variable y_{ks} dans la fonction objectif du problème maître. Le coût réduit de la colonne A_{ks} peut s'écrire $\overline{W}_{ks} = W_{ks} - \lambda \cdot A_{ks}$. Le poids W_{ks} comprend le salaire g_{ks} et le niveau de préférence f_{ks} de l'infirmière, de même qu'un troisième terme correspondant à la préférence (du décideur) pour son niveau de qualification:

$$W_{ks} = g_{ks} - f_{ks} - \sum_{d\in\mathcal{D}}\sum_{p\in\mathcal{P}}\sum_{\ell=1}^{L}\sum_{t=1}^{T} e^{\ell}_{pd}\alpha_{tp}b_{k\ell}a_{std}.$$

Les coûts sur les arcs sont déduits en développant l'expression de $\overline{W}_{ks}$ et en la posant égale au coût $\sum_{(v_i,v_j)\in A} c_{ij}z_{ij}$ du chemin associé à la colonne A_{ks}, où c_{ij} est le coût sur l'arc (v_i, v_j) et z_{ij} est égale à 1 si l'arc (v_i, v_j) est sur le chemin et 0 sinon.

Le salaire de l'infirmière k comprend un salaire de base g^0_k et une rémunération supplémentaire g^j_k pour l'affectation au quart t_j le jour d_j. Le calcul de f_{ks} implique un ajustement selon une mesure, γ_k, de l'ancienneté de l'infirmière k, ainsi qu'une réduction par une quantité f^0_k proportionnelle au niveau de préférence de son horaire pour l'horizon précédent. Ceci permet d'incorporer une certaine équité dans la génération des horaires d'un horizon à l'autre. Soit f^j_k, la préférence de l'infirmière k pour le quart de travail t_j le jour d_j, on a:

$$\begin{aligned} g_{ks} &= g^0_k + \sum_{(v_i,v_j)\in A} g^j_k z_{ij}, \\ f_{ks} &= \sum_{(v_i,v_j)\in A} \gamma_k f^j_k z_{ij} \quad - \quad f^0_k. \end{aligned}$$

Soit un sommet $v_{j'}$ tel que $d_{j'} = d$ et $t_{j'} = t$, alors, étant donné un horaire s, on a:

$$a_{std} = \sum_{(v_i,v_{j'})\in A} z_{ij'}. \tag{5.5}$$

Par conséquent, le terme correspondant à la préférence pour le niveau de qualification de l'infirmière peut se récrire:

$$\sum_{(v_i,v_j)\in A}\sum_{p\in\mathcal{P}}\sum_{\ell=1}^{L} e^{\ell}_{pd_j}\alpha_{t_jp}b_{k\ell}z_{ij} \ .$$

Considérons maintenant C, l'ensemble des quadruplets $(p, d, \mathcal{L}, \mathcal{T})$ représentant les contraintes de demande (5.3) du problème maître. Soient $\lambda_{pd}^{\mathcal{LT}}$ la variable duale associée à un certain élément de C et λ_k la variable duale correspondant à la k-ème contrainte (5.2), alors:

$$\lambda . A_{ks} = \lambda_k + \sum_{(p,d,\mathcal{L},\mathcal{T})\in C} \sum_{\ell\in\mathcal{L}} \sum_{t\in\mathcal{T}} \alpha_{tp} b_{k\ell} a_{std} \lambda_{pd}^{\mathcal{LT}}.$$

En utilisant (5.5) et en introduisant le paramètre

$$\beta_{\mathcal{T}d}^{td'} = \begin{cases} 1 & \text{si } t \in \mathcal{T} \text{ et } d' = d \\ 0 & \text{sinon,} \end{cases}$$

on obtient:

$$\lambda^T A_{ks} = \lambda_k + \sum_{(v_i,v_j)\in A} \sum_{(p,d,\mathcal{L},\mathcal{T})\in C} \sum_{\ell\in\mathcal{L}} \alpha_{t_j p} b_{k\ell} \beta_{\mathcal{T}d}^{t_j d_j} \lambda_{pd}^{\mathcal{LT}} z_{ij}.$$

Finalement, en posant le coût réduit $\overline{W}_{ks}$ égal à la longueur du chemin associé à A_{ks}, on a les longueurs d'arcs:

- pour les arcs (v_1, v_j) quittant la source:

$$\begin{aligned} c_{1j} &= f_k^0 + g_k^0 - \lambda_k + g_k^j - \gamma_k f_k^j \\ &\quad - \sum_{p\in\mathcal{P}} \sum_{\ell=1}^{L} e_{pd_j}^{\ell} \alpha_{t_j p} b_{k\ell} - \sum_{(p,d,\mathcal{L},\mathcal{T})\in C} \sum_{\ell\in\mathcal{L}} \alpha_{t_j p} b_{k\ell} \beta_{\mathcal{T}d}^{t_j d_j} \lambda_{pd}^{\mathcal{LT}}, \end{aligned}$$

- pour tous les autres arcs (v_i, v_j):

$$\begin{aligned} c_{ij} &= g_k^j - \gamma_k f_k^j - \sum_{p\in\mathcal{P}} \sum_{\ell=1}^{L} e_{pd_j}^{\ell} \alpha_{t_j p} b_{k\ell} \\ &\quad - \sum_{(p,d,\mathcal{L},\mathcal{T})\in C} \sum_{\ell\in\mathcal{L}} \alpha_{t_j p} b_{k\ell} \beta_{\mathcal{T}d}^{t_j d_j} \lambda_{pd}^{\mathcal{LT}}. \end{aligned}$$

5.3 Résolution du problème maître

5.3.1 L'arbre de branchement

On utilisera un schéma d'énumération implicite combiné avec la méthode de génération de colonnes pour résoudre le problème maître. Il s'agit d'un problème de grande taille: le nombre de variables est égal à celui d'horaires individuels potentiels (i.e, au nombre de chemins dans le réseau) multiplié par le nombre d'employés.

Une abondante littérature existe sur l'utilisation de la génération de colonnes pour résoudre les programmes linéaires de grande taille en variables entières ou mixtes (voir, par exemple, Barnhart *et al.* [6], Hansen *et al.* [37] ou Vanderbeck et Wolsey [61]). Si une méthode d'énumération implicite est utilisée pour résoudre le problème, il essentiel de définir des sous-problèmes dont la structure soit compatible avec celles du problème maître et du problème auxiliaire.

Le sous-problème, à un noeud donné de l'arbre d'énumération, comprend le problème maître initial et une série de contraintes de branchement. La relaxation continue de ce sous-problème est résolue par génération de colonnes. En d'autres termes, cette relaxation est résolue à l'optimalité, par la méthode du simplexe, pour seulement un sous ensemble de colonnes (horaires potentiels), afin d'obtenir des variables duales qui sont ensuite introduites dans le problème auxiliaire pour générer de nouvelles colonnes de coûts réduits négatifs, incluant, de préférence, celle de coût réduit minimal. S'il n'existe pas de colonne de coût réduit négatif, la solution courante est optimale pour la relaxation continue du sous-problème.

De nouvelles branches (i.e. sous-problèmes) sont créées si la solution est réalisable mais fractionnaire et de plus petite valeur que la meilleure solution entière connue. Si par contre la solution est entière, elle peut être utilisée pour mettre à jour la meilleure solution entière connue. Un nouveau noeud est ensuite sélectionné et le processus recommence, jusqu'à ce que l'arbre soit entièrement exploré.

5.3.2 La règle de branchement

Le branchement peut se faire sur les variables du problème maître ou sur celles du problème auxiliaire, mais en pratique, il est souvent plus efficace de brancher sur les variables du problème auxiliaire (voir, par exemple, Barnhart *et al.* [6] ou Desrosiers *et al.* [22]). Le schéma de branchement considéré ici consiste à forcer ou à éviter sélectivement certaines affectations pour certaines infirmières, en fixant ou en interdisant certains sommets dans les problèmes auxiliaires appropriés.

Tous les arcs (i.e., variables du problème auxiliaire), arrivant ou quittant les sommets spécifiés par les contraintes de branchement actives pour le sous-problème courant, sont par conséquent imposés ou éliminés par cette règle de branchement. Ainsi, plusieurs horaires potentiels (ou variables du problème maître), contenant les affectations spécifiées, sont affectés par chacune des deux décisions de branchement à un noeud donné.

La configuration optimale d'horaires pour l'ensemble du personnel est ainsi construite en fixant progressivement les affectations dans les horaires individuels. On peut remarquer que le nombre de variables de branchement, i.e. le nombre total de sommets, est $O(KT|\mathcal{D}|)$, puisqu'il y a T quarts de travail et $|\mathcal{D}|$ jours, et donc au plus $T|\mathcal{D}|$ sommets dans le graphe de la k-ème infirmière. Ce nombre est relativement limité, en comparaison avec les $O(KT^{|\mathcal{D}|})$ horaires potentiels qu'il faudrait considérer pour un branchement sur les variables du problème maître.

Le sommet sur lequel le branchement doit s'effectuer, peut être choisi en examinant les affectations dans les horaires correspondant aux variables fractionnaires de la solution optimale de la relaxation linéaire du sous-problème courant. En effet, soit Y_k, l'ensemble des variables fractionnaires (i.e., horaires) y_{ks} associées à l'infirmière k. Considérons l'ensemble V_k des affectations (i.e., sommets) que l'infirmière k reçoit dans au moins un de ses horaires correspondant à une variable fractionnaire, i.e., $V_k = \{v_j \in G_k : a_{t_j d_j s} = 1,\ y_{ks} \in Y_k\}$, où G_k est le graphe associé à l'infirmière k. Définissons le poids d'un sommet de G_k comme la somme des valeurs des variables $y_{ks} \in Y_k$ dans lesquelles apparaît l'affectation correspondant au sommet.

Il est clair que le poids d'un sommet appartient à l'intervalle]0, 1], du fait des contraintes de partitionnement (5.2) dans le problème maître. Une règle de branchement possible consiste à choisir le sommet v^*, dont le poids est le plus proche de 0.5 et différent de l'unité, tel que donné ci-après.

Théorème 5.1 *Si, à chaque itération, le branchement s'effectue sur un sommet* $v^* = argmin\{\left|\sum_{y_{ks} \in Y_k} a_{t_j d_j s} y_{ks} - 0.5\right| : \sum_{y_{ks} \in Y_k} a_{t_j d_j s} y_{ks} \neq 1,\ v_j \in V_k,\ k = 1, 2, ..., K\}$, *alors l'arbre d'énumération implicite sera exploré en un nombre fini d'étapes.*

Preuve. On montre d'abord que le sommet v^* ne peut être impliqué dans aucune des contraintes de branchement associées au sous-problème courant. Supposons, par l'absurde, que v^* a déjà été sélectionné dans un sous-problème dont descend le sous-problème courant. Alors, soit l'affectation correspondant à v^* n'apparaît dans aucun des horaires fractionnaires que la solution optimale courante attribue à l'infrmière k^* associée à v^*, soit l'affectation apparaît dans tous ces horaires (le branchement est binaire).

En d'autres termes, le poids $\sum_{y_{ks} \in Y_k} a_{t^* d^* s} y_{ks}$ est égal à 0 ou à 1. Dans le premier cas, v^* n'appartient pas à V_{k^*} et donc ne peut être sélectionné par la règle énoncée. Dans le deuxième cas, la condition $\sum_{y_{ks} \in Y_k} a_{t_j d_j s} y_{ks} \neq 1$ est violée et v^* ne peut, non plus, être candidat au branchement à partir du sous-problème courant.

Ainsi donc, on ne peut regénérer une contrainte de branchement déjà active dans le sous-problème courant, ni le complémentaire de cette contrainte. On ne peut non plus générer une solution violant une contrainte de branchement active pour le sous-problème courant, puisque les chemins sont déterminés dans le problème auxiliaire en évitant les sommets interdits et en incluant obligatoirement les sommets fixés. Comme le nombre de sommets dans le graphe associé à chaque infirmière est fini, le résultat suit.□

Notons que cette règle de branchement implique très peu de modifications du problème auxiliaire de plus court chemin avec fenêtres de ressource. Si un sommet donné est imposé par une contrainte de branchement, alors $-M$, où M est une

valeur suffisamment grande, est ajouté au coût des arcs quittant ce sommet. Une valeur possible pour M est $2n$ fois la plus grande valeur absolue des coûts sur les arcs du graphe initial, où n est le nombre de sommets.

Si le sommet est plutôt interdit, les coûts sur les arcs correspondants sont remplacés par $+\infty$. Le problème auxiliaire sera ensuite résolu pour trouver un chemin dont le coût est inférieur à $-M$ fois le nombre de sommets imposés. Ceci peut se faire en utilisant le même algorithme qu'avant le branchement (i.e. la phase 2 de l'algorithme en deux phases) et en ne traitant que les sommets non interdits. Nous discutons, dans la prochaine section, une implantation de cette procédure de résolution.

5.4 Implantation

5.4.1 Méthodologie

Traitement du problème maître

En pratique, il est généralement difficile de trouver une solution réalisable pour le problème d'horaires d'infirmières, à cause de la nature conflictuelle des contraintes devant être considérées et de la taille du problème. L'implantation effectuée privilégie donc une recherche rapide de solutions réalisables entières, afin de pouvoir arrêter, éventuellement, la résolution sans avoir nécessairement obtenu une solution optimale. La solution de la relaxation continue du problème permet cependant de calculer le saut d'intégralité, ce qui fournit une borne sur les améliorations éventuellement possibles.

Une technique de recherche en profondeur d'abord est donc utilisée pour explorer l'arbre de branchement, puisque cette méthode permet de trouver plus rapidement une solution réalisable entière et requiert moins de mémoire qu'une recherche par le

meilleur d'abord. En outre, une recherche de solutions entières réalisables se fait durant le processus de génération de colonnes, lors de la résolution de la relaxation continue d'un sous-problème de branchement.

En effet, la solution optimale de la restriction de ce problème à un sous-ensemble de colonnes, i.e. la solution du problème résolu entre deux appels du problème auxiliaire, peut être entière. Elle peut donc éventuellement servir à mettre à jour la liste des solutions entières retenues ou même la meilleure solution entière connue.

Notons que l'utilisation de la méthode de la génération de colonnes permet de sélectionner les infirmières pour qui de nouveaux horaires seront générés lors d'un l'appel du problème auxiliaire. Cependant, cette option n'a pas été implantée. Dans tous les tests présentés ici, la liste des infirmières est parcourue d'une manière cyclique jusqu'à ce qu'au moins une colonne de coût réduit négatif soit trouvée.

En général, plusieurs de ces colonnes sont générées lorsque le problème auxiliaire est résolu pour une infirmière donnée. Toutes ces colonnes sont introduites dans le problème maître si cela est possible sans toucher à la base optimale courante. L'implantation prend également en compte des variables de déficit et de surplus, avec des pénalités élevées, pour éviter la non-réalisabilité tout en minimisant les violations de quotas.

Un problème rencontré en pratique avec la méthode de la génération de colonnes est la dégénérescence de la base optimale de la relaxation linéaire du problème maître restreint à un sous-ensemble de colonnes. Dans ce cas, l'introduction de nouvelles colonnes de coûts réduits négatifs ne se traduit pas toujours par l'amélioration de la fonction objectif. Une méthode de stabilisation est décrite dans Du Merle *et al.* [26] pour ce problème.

Un traitement sommaire du problème a été implanté dans le cadre de cette thèse. Il consiste à brancher lorsqu'après un certain nombre d'appels du problème auxiliaire, la valeur de la solution reste pratiquement inchangée. Après un certain nombre, fixé *a priori*, de ces branchements éventuellement sous-optimaux, la solution obtenue est considérée, comme optimale pour le sous-problème correspondant.

Une façon alternative de trouver rapidement une configuration d'horaires serait de développer une heuristique pour arrondir la solution de la relaxation linéaire du problème maître initial. Cette approche, qui est susceptible de donner des solutions non réalisables par rapport aux contraintes de demande, pourrait être considérée lorsque les surplus et/ou les déficits de quotas ne sont pas critiques.

Traitement du problème auxiliaire

Dans le but de réduire l'espace mémoire requis par les différentes instances du problème auxiliaire, les infirmières devant recevoir sensiblement les mêmes horaires, sont regroupées de manière à former un profil-type auquel est associé un seul graphe. Ces profils sont tels que la structure des ressources utilisées comprend, comme cas particulier, les spécifications relatives à chaque personne impliquée. Cela permet de réduire le nombre de phases 1 de l'algorithme.

Les particularités individuelles sont prises en compte durant la phase 2. A la fin de cette dernière, les chemins dont les vecteurs de ressource ne satisfont pas les spécifications propres à la personne, considérée sont rejetés ou des pénalités supplémentaires leur sont appliquées selon leur déviation.

L'implantation permet ainsi d'accepter éventuellement (comme c'est le cas en pratique) des solutions avec de légères déviations par rapport aux spécifications des contraintes de la convention collective. Les particularités (i.e. déviations acceptables) relatives au traitement des congés de fins de semaines sont prises en compte d'une façon similaire. De telles situations apparaissent, par exemple, lorsque l'infirmière reçoit un certain nombre de jours de vacance et/ou de congé statutaire pendant la semaine précédant ou suivant une fin de semaine donnée, tout en travaillant, éventuellement, les autres jours.

Un autre artifice utilisé dans l'implantation consiste à prendre, comme unité pour la ressource correspondant à la charge de travail, le plus grand diviseur commun des durées de quarts de travail. Cela permet de réduire le nombre de valeurs possibles pour cette ressource et donc le nombre total de vecteurs de ressource à examiner.

5.4.2 Description des données

Les tests on été effectués sur des données en provenance du Centre des Naissances de l'Hôpital Royal Victoria de Montréal. L'horizon considéré est de 28 jours (4 semaines) pour un effectif de 54 infirmières, comprenant le personnel permanent et le personnel flottant basé à l'unité. Douze niveaux de qualifications et sept quarts de travail sont considérés.

Les facteurs considérés dans la génération des horaires individuels concernent la charge de travail, les fins de semaines, les rotations, les ratios de quarts de jour ou du soir (pour les personnes recevant uniquement des affectations du soir et de nuit). Un total de 27 profils d'infirmières (i.e. de graphes différents pour le problème auxiliaire) a été utilisé. Le tableau 5.1 donne des statistiques (moyenne, écart-type et valeurs extrêmes) sur le nombre de sommets, le nombre d'arcs et le nombre de ressources pour ces profils.

Tableau 5.1 – *Statistiques du prétraitement des profils*

	Moyenne	Écart-type	Min	Max
Temps CPU (sec)	18.300	24.575	0.020	91.960
Nb. de sommets	50.259	24.062	15	114
Nb. d'arcs	1776.037	1973.677	155	9469
Nb. de ressources	4.593	0.888	2	5

Les quotas sont spécifiés pour quatre périodes par jour et donnent le nombre total de personnes devant être présentes pour chaque période et pour chaque jour de l'horizon, sans distinction de quarts de travail ou de qualification. Quelques contraintes supplémentaires de demande sont cependant spécifiées pour quelques jours, afin d'assurer la présence d'un certain nombre de personnes avec des qualifications définies. Le programme maître initial contient, typiquement, environ 150 contraintes.

Dans les tests effectués, seuls les préférences sont prises en compte dans la fonction objectif, i.e. un poids nul est accordé au coût salarial. Les valeurs de préférences

individuelles pour des affectations spécifiques varient entre 0 et 5. Une personne peut recevoir entre 4 et 19 affectations par horaire, selon les règles de la convention collective qui s'appliquent à elle.

Des pénalités de 10^5 et de 10^4 sont respectivement associées aux variables de surplus et de déficit des quotas de présences. Des pénalités de 10 à 100 fois plus élevées sont utilisées pour les violations des contraintes de ressource spécifiques à chaque individu, puisque de tels écarts sont moins tolérés en pratique.

Les fenêtres de ressource d'étendues minimales et maximales sont données aux tableaux 5.2 et 5.3 pour les graphes (profils) correspondant respectivement aux temps de calcul minimal et maximal. Les spécifications minimales et maximales pour les ratios de quarts de travail de jour (ou du soir) sont respectivement de 0.4 et 0.6 par rapport au nombre total d'affectations dans l'horaire individuel considéré.

Tableau 5.2 – *Étendues des fenêtres pour le profil de plus petite taille*

Ressources	Seuils de mise à jour		Val. de mise à jour		Bornes de réalisab.	
	Min	Max	Min	Max	Min	Max
Charges de travail	[0, 0]	[2, 12]	[0, 0]	[2, 12]	[0, 0]	[2, 10]
Rotation	[1, 4]	[0, 4]	[1, 4]	[0, 4]	[4, 4]	[0, 4]

Tableau 5.3 – *Étendues des fenêtres pour le profil de plus grande taille*

Ressources	Seuils de mise à jour		Val. de mise à jour		Bornes de réalisab.	
	Min	Max	Min	Max	Min	Max
Charges de travail	[0, 0]	[2, 20]	[0, 0]	[2, 20]	[0, 0]	[2, 18]
Fins de semaines	[1, 1]	[-1, 1]	[-1, -1]	[-1, -1]	[1, 1]	[-1, 1]
Rotation	[1, 3]	[0, 9]	[1, 3]	[0, 9]	[0, 0]	[0, 9]
Ratio de jours	[0, 0]	[0, 28]	[0, 0]	[0, 28]	[0, 0]	[0, 11]
Ratio de soirs-nuits	[0, 0]	[0, 28]	[0, 0]	[0, 28]	[0, 0]	[0, 11]

5.4.3 Analyse des surplus et déficits d'affectations

Les tableaux 5.4 et 5.5 donnent les surplus et les déficits des quotas et des consommations de ressource pour quelques solutions obtenues. Il s'agit respectivement d'une solution de la relaxation linéaire du problème maître, des quatre meilleures solutions entières obtenues après 1 heure 22 minutes de calcul et de la solution manuelle obtenue par l'infirmière-chef. Ces déviations sont calculées sur tout l'horizon et pour l'ensemble du personnel.

Tableau 5.4 – *Surplus des quotas et des ressources*

	Relax. lin.	Sol.1	Sol.2	Sol.3	Sol.4	Sol. man.
Quotas	40	40	40	40	40	52
Charg. de trav. (heures)	0	0	0	0	0	0
Congés de fins de sem.	4	8	7	5	7	11
Affect. consécutives	0	0	0	0	0	0
Ratios de jours (somme)	0	0	0	0	0	3.4
Ratios de soirs (somme)	0	0	0	0	0	0

Tableau 5.5 – *Déficits des quotas et des ressources*

	Relax. lin.	Sol.1	Sol.2	Sol.3	Sol.4	Sol. man.
Quotas	0	0	0	0	0	11
Charg. de trav. (heures)	0	0	0	0	0	0
Congés de fins de sem.	0	0	0	0	0	0
Affect. consécutives	8	90	95	98	92	80
Ratios de jours (somme)	0	0	0	0	0	1.37
Ratios de soirs (somme)	0	0	0	0	0	0.26

La solution optimale (fractionnaire) de la relaxation linéaire indique un surplus de 40 et un déficit de 0 présences pour l'ensemble du personnel durant l'horizon. Cela signifie que, compte tenu des contraintes de la convention collective, de la taille du personnel et des poids accordés aux préférences exprimées et aux violations de quotas, il est impossible d'éliminer tous les surplus d'affectations. Les mêmes violations de quotas sont observées pour les (meilleures) solutions entières obtenues après 1 heure 22 minutes de calcul. Ces écarts restent cependant moins élevées que les valeurs obtenues à la main.

On note un surplus de congés de fins de semaines pour toutes les solutions obtenues. Ces congés supplémentaires ne constituent pas nécessairement un inconvénient. Les pénalités relatives aux surplus de fins de semaines ont donc été réduites d'un facteur de 10 par rapport à celles correspondant aux déficits. Ces derniers sont difficilement tolérés en pratique, à moins d'être explicitement demandés.

Les déficits par rapport aux nombres minima d'affectations consécutives avant congé (ou avant rotation) sont cependant élevés. Dans la plupart des cas, ces nombres représentent des quarts de travail isolés, i.e. précédés et suivis immédiatement d'au moins un jour de congé. On peut éliminer ces déficits, si nécessaire, en fixant le nombre minimal d'affectations consécutives à 2 au moins.

La somme totale (sur l'ensemble du personnel) des différences entre les ratios de quarts de travail et les limites spécifiées, est nulle, contrairement à ce que donne la solution manuelle.

5.4.4 Analyse des performances techniques du modèle

Le programme a été codé en langage C. Les fonctions de la bibliothèque du logiciel commercial CPLEX sont utilisées pour la résolution et la mise à jour de la relaxation linéaire du problème maître restreint à un sous-ensemble de colonnes. Ces fonctions sont appelées plusieurs fois à chaque noeud de branchement. Aucun autre logiciel ou bibliothèque de fonctions n'a été utilisé.

Les tests ont été faits sur une station de travail Sun Ultra 2 et ont nécessité environ 95 mégaoctets d'espace mémoire en tout. L'essentiel de cet espace mémoire a été nécessaire durant la phase 1 de l'algorithme de plus court chemin, exécutée au début de la résolution du problème, en vue de caractériser les horaires admissibles.

Le tableau 5.6 donne les caractéristiques techniques des solutions obtenues (saut de dualité, temps de calcul et nombre de noeuds de branchement). On remarque que toutes les solutions entières présentées sont obtenues à un même noeud de branchement. Cela signifie que certaines solutions du problème résolu entre deux appels du problème auxiliaire sont entières, pour le sous-problème associé à ce noeud. Ces solutions entières presqu'optimales (saut de dualité d'environ 0.01%) ont été obtenues en 1 heure 22 minutes de calcul environ et après seulement 125 noeuds de branchement.

Tableau 5.6 – *Caractéristiques techniques des solutions*

	Relax. lin.	Sol.1	Sol.2	Sol.3	Sol.4
Saut dual. (%)	0.00	0.01	0.01	0.01	0.01
Temps CPU (sec)	612	4872	4869	4874	4869
Nb. noeuds de branch.	0	125	125	125	125

Deux autres solutions ayant pratiquement les mêmes caractéristiques ont été obtenues au même noeud. Nous avons laissé fonctionner le programme pendant environ 45 minutes supplémentaires mais aucune nouvelle solution entière n'a été trouvée. L'exploration de l'arbre de branchement a ensuite été interrompue du fait de la qualité des solutions obtenues et du caractère plutôt subjectif des préférences considérées dans la fonction objectif.

Nous pensons que les temps de calcul obtenus peuvent être encore améliorés, par un meilleur traitement de la dégénérescence de la relaxation linéaire des sous-problèmes de branchement. Dans l'implantation décrite ici, un branchement est effectué si, après avoir parcouru la liste des infirmières, les colonnes (horaires) de coûts réduits négatifs générées ne permettent pas une amélioration de l'objectif de 0.01% au noeud initial et de 1% à tous les autres noeuds.

Si la même situation se reproduit à un noeud issu d'un tel branchement, la meilleure solution obtenue à ce noeud est considérée comme optimale pour la relaxation linéaire du sous-problème correspondant. Un meilleur contrôle des paramètres relatifs à ce traitement de la dégénérescence pourrait réduire le temps de calcul.

Tableau 5.7 – *Statistiques des sous-problèmes*

	Moyenne	Écart-type	Min	Max
Temps CPU (sec)	38.235	15.927	15.250	116.200
Nb. de PM	64.945	29.570	21	196
Nb. de PA	83.749	33.928	31	222
Nb. de colonnes	971.066	479.365	293	4548

Le tableau 5.7 donne des statistiques sur les sous-problèmes de branchement. En moyenne, la résolution de la relaxation linéaire d'un tel sous-problème requiert un temps de calcul de 38.235 secondes, environ 65 résolutions du problème maître (PM) restreint à un sous-ensemble de colonnes et 84 appels du problème auxiliaire (PA) qui génèrent environ 971 colonnes.

Tableau 5.8 – *Temps de calcul du problème maître et du problème auxiliaire*

	Moyenne	Écart-type	Min	Max
PM: temps CPU (sec)	0.025	0.037	0.000	0.530
PA: temps CPU (sec)	0.374	0.622	0.000	2.940
Nb. de colonnes	12.614	24.202	0	122

Le tableau 5.8 permet une comparaison des temps de calcul requis par la résolution de la relaxation linéaire du problème maître restreint (par CPLEX) et la phase 2 de l'algorithme proposé. Rappelons que ce dernier a été légèrement modifié pour tenir compte du fait que certains chemins réalisables, pour un profil donné, peuvent s'écarter légèrement des spécifications propres à une personne particulière.

Le programme linéaire (PM) est résolu, à l'aide du logiciel CPLEX, en 0.025 secondes, en moyenne, tandis que la résolution du problème auxiliaire (PA) requiert 0.374 secondes. Chaque résolution du problème auxiliaire produit, en moyenne, environ 13 colonnes de coûts réduits négatifs.

On peut noter que les temps correspondant à la phase 2 sont relativement plus élevés que ceux observés au chapitre 4 (tableau 4.2). Cela s'explique par les modifications requises par l'utilisation d'un même graphe (profil) pour plusieurs infirmières. Ces temps restent cependant nettement plus faibles que ceux de la phase 1 (tableau 5.1), qui requiert en moyenne 18.300 secondes.

5.4.5 Quelques commentaires

Le modèle de programmation linéaire généralisé en variable 0-1 développé dans ce chapitre permet de prendre en compte la plupart des contraintes rencontrées en pratique, dans la confection d'horaires de personnel soignant. En particulier, toutes les contraintes identifiées lors des discussions à l'Hôpital Royal Victoria (voir la référence [15]) ont été modélisées et testées numériquement.

L'approche utilisée permet une exploration implicite complète de l'ensemble des horaires potentiels. Un avantage majeur de cette méthode par rapport à une approche heuristique est sa flexibilité d'une unité de soin à l'autre. Très peu d'ajustements seront nécessaires lors de ces changements d'environnement.

A notre connaissance, ce travail constitue la première tentative de résolution exacte d'un modèle réaliste et flexible pour le problème d'horaires de personnel soignant. Le modèle peut être utilisé, moyennant quelques modifications mineures, pour la plupart des problèmes d'horaires de personnel dans les organisations opérant en continu.

Une attention particulière est cependant requise lors de la définition des fenêtres de ressource et des valeurs de mise à jour, dans le problème auxiliaire. Il faut, en

effet, tenir compte du fait que la complexité de l'algorithme de plus court chemin avec fenêtres de ressource est très sensible à la largeur des fenêtres.

Un système automatisé, basé sur le modèle proposé requiert des données statiques et dynamiques. Les données statiques sont relatives aux contraintes de demande et de la convention collective, aux caractéristiques des quarts de travail ainsi qu'aux informations individuelles de base nécessaires à la confection d'horaires personalisés. Très peu de données dynamiques seront requises pour chaque horizon. Elles serviront notamment à spécifier les requêtes et les préférences. Si ces dernières ne sont pas données le système peut être automatiquement initialisé avec des valeurs par défaut prédéfinies.

Puisqu'un tel système ne génère que des solutions réalisables (s'il en existe), sur la base d'une minimisation des coûts et/ou d'une maximisation des préférences, très peu d'ajustements seront nécessaires, par la suite, de la part de l'infirmière-chef. Notons que si le système ne trouve pas de solution réalisable en résolvant la relaxation linéaire, cela signifie qu'une telle solution n'existe pas pour les contraintes spécifiées.

Cette méthode requiert cependant des ressources importantes en espace mémoire et en temps de calcul (quelques heures pour trouver des solutions utilisables en pratique, après éventuellement quelques modifications mineures). L'espace mémoire nécessaire peut être réduite par une définition appropriée des profils d'infirmières, par exemple, en utilisant une heuristique pour évaluer préalablement divers regroupements des infirmières. Les temps de calcul élevés ne constituent pas un inconvénient majeur, étant donné que l'horizon considéré est de 28 jours pour les tests effectués (6 semaines en pratique).

Notons enfin que l'utilisation du système ne résultera pas nécessairement en une réduction du personnel. Sa capacité à prendre en compte la plupart des contraintes rencontrées en pratique et de générer éventuellement plusieurs (bonnes) solutions réalisables aura cependant un impact positif direct sur la qualité des soins. Le temps passé par l'infirmière-chef à confectionner les horaires en sera considérablement réduit.

CONCLUSION

Nous avons étudié dans cette thèse différents problèmes de cheminement impliquant deux objectifs ou des contraintes de ressources. Il s'agit, spécifiquement, des problèmes de chemins avec étendue ou ratio minimum, du problème de plus court chemin bicritère et du problème de plus court chemin avec fenêtres de ressources. Une analyse détaillée du problème d'horaires de personnel soignant a également été faite à titre d'application du problème de plus court chemin avec fenêtres de ressource. Ce travail a permis d'apporter diverses contributions à la formulation et à la résolution des problèmes considérés.

Des algorithmes polynomiaux ont été notamment développés pour de nouveaux problèmes de chemins avec étendue ou ratio minimum pouvant apparaître, par exemple, comme sous-problèmes en équilibrage des chaînes de montage lorsqu'on s'intéresse à la minimisation du nombre de postes de travail et/ou du temps d'attente dans le pire cas. En particulier, une procédure a été donnée pour déterminer tous les chemins efficaces du problème bicritère de plus court chemin avec étendue minimale. Ce problème bicritère, qui n'avait jamais été étudié auparavant, considère la minimisation de la longueur totale du chemin et la différence entre la plus grande longueur d'arc et la plus petite.

Un nouvel algorithme d'étiquetage a également été proposé pour résoudre le problème de plus court chemin bicritère (à coûts non négatifs) par les deux extrémités du réseau. Ce problème se rencontre, par exemple, en transport de matières dangereuses si l'on minimise à la fois le coût de transport et la population exposée.

L'algorithme se compare favorablement à la méthode de résolution par une seule extrémité, lorsque la taille ou la densité du graphe augmente. La procédure peut aisément être modifiée pour traiter les cas où des bornes sont imposées sur la valeur de chacun des deux critères, afin d'éviter les solutions impliquant une trop grande détérioration de l'un d'eux.

Une nouvelle formulation, plus générale, du problème de plus court chemin avec fenêtres de ressources, a été proposée, de même que divers algorithmes pseudo-polynomiaux, dont une procédure spécialisée, en deux phases. Cette dernière ne nécessite pas un dédoublement des ressources lorsqu'aucune déviation n'est permise par rapport aux bornes des fenêtres inférieures. L'extension des consommations de ressource n'est pas restreinte non plus à des fonctions non-décroissantes.

Cette généralisation ne détériore cependant pas la complexité de l'algorithme. Les calculs effectués indiquent en outre que la borne donnée dans la littérature pour la résolution du problème de plus court chemin avec fenêtres de temps aux sommets est sur-évaluée. La structure en deux phases de l'algorithme proposée convient particulièrement à la réoptimisation, après une modification des coûts sur les arcs, comme c'est le cas lorsque le problème apparaît dans un processus de génération de colonnes.

D'un point de vue plus appliqué, une formulation de plus court chemin avec fenêtres de ressource a été proposée et testée avec succès, pour le problème de génération d'horaires réalisables pour une infirmière donnée. Il s'agit d'un problème pratique et complexe de cheminement, nécessitant de fréquentes réoptimisations, qui peut être traité plus efficacement par l'algorithme en deux phases que par les autres algorithmes de plus court chemin disponibles dans la littérature. Le modèle proposé est réaliste et capable de tenir compte de la plupart des règles de la convention collective, utilisées en pratique pour construire des horaires personalisés.

Un modèle de programmation linéaire généralisée, en variables 0-1, a été développé, pour le problème, plus général, de la confection d'horaires pour l'ensemble du personnel soignant d'une unité. Le modèle est susceptible de résolution exacte par une procédure d'énumération implicite combinée à la génération de colonnes.

Ce dernier modèle contient, comme problème auxiliaire, celui de la génération d'horaires réalisables pour une infirmière donnée. Il permet également une exploration complète de l'ensemble des horaires potentiels, contrairement à la plupart des formulations de la littérature où l'on considère des horaires cycliques ou prédéfinis et en nombre relativement limité. Une règle de branchement spécialisée a été proposée pour la résolution du problème et sa convergence prouvée.

Un prototype du modèle a été implanté en langage C, en utilisant les fonctions de la bibliothèque de l'optimiseur commercial CPLEX pour résoudre et mettre à jour les sous-problèmes linéaires. Le prototype a été testé avec satisfaction, sur des données réelles en provenance de l'Hôpital Royal Victoria de Montréal. De l'avis des infirmières-chefs, les horaires générés par le programme sont de qualité au moins égale à celle des horaires produits à la main.

Des profils-types d'infirmières ont été définis afin de réduire l'espace mémoire requis, en regroupant les personnes devant recevoir sensiblement les mêmes types d'horaires. Cependant, une implantation plus efficace peut être obtenue en utilisant une heuristique pour trouver les profils-types. Un meilleur compromis pourrait ainsi être trouvé entre la réduction de la mémoire requise et l'augmentation du temps de calcul, due à l'utilisation d'un même réseau pour plusieurs infirmières.

Par ailleurs, une heuristique peut également être développée pour arrondir la solution de la relaxation continue du problème maître, étant donnée qu'en pratique de petites déviations sont acceptées par rapport aux quotas. Une telle heuristique sera notamment utile si la solution de la relaxation linéaire contient des déficits ou des surplus par rapport aux quotas.

RÉFÉRENCES

[1] APPELGREN, L.H. (1969). A column generation algorithm for a ship scheduling problem. *Transportation Science*, 3, 53-68.

[2] AHUJA, R.K. (1988). Minimum cost-reliability ratio path problem. *Computers and Operations Research*, 15 (1), 83-89.

[3] ANEJA, Y.P., AGGARWAL, V. et NAIR, K.P.K. (1983). Shortest chain subject to side constraints. *Networks*, 13, 295-302.

[4] ANZAI, M. et MIURA, Y. (1987). Computer program for quick work scheduling of nursing staff. *Medical Information*, 12 (1), 43-52.

[5] BAKER, K. (1974). Scheduling a full-time workforce to meet cyclic staffing requirements. *Management Science*,20 (12), 1561-1568.

[6] BARNHART, C., JOHNSON, E.L., NEMHAUSER, G.L. SAVELSBERGH, M.W.P. et VANCE, P.H. (1994). Branch-and-Price: column generation for solving huge integer programs. *Mathematical Programming: State of the Art*, J.R. Birge et K.G.Murty, (Éditeurs). The University of Michigan, USA.

[7] BAYBARS, I. (1986). A survey of exact algorithms for the simple assembly line balancing problem. *Management Science*, 32, 109-132.

[8] BEASLEY, J.E. et CHRISTOFIDES, N. (1989). An algorithm for the sesource constrained shortest path problem. *Networks*, 19, 379-394.

[9] BERRADA, I. (1993). *Planification d'horaires du personnel infirmier dans un établissement hospitalier*. Thèse de doctorat (Ph.D.) en Informatique, Université de Montréal

[10] BURNS, R.N. (1978). Manpower scheduling with variable demands and alternate weekends off. *INFOR* 16(2), 101-111.

[11] BURNS, R.N. et KOOP, G.J. (1987). A modular approach to optimal multiple-shift manpower scheduling. *Operations Research*, 35(1), 100-110.

[12] CHIN, S.M. et CHENG, P. D. M. (1989). Bicriterion routing scheme for nuclear spent fuel transportation. *Transportation Research Record*, 1245, 60-64.

[13] CLIMACO, J.C.N. et MARTINS, E.Q.V. (1982). A bicriterion shortest path algorithm. *European Journal of Operational Research*, 11, 399-404.

[14] CURRENT, J. et MARSH, M. (1993). Multiobjective transportation network design and routing problems: taxonomy and annotation. *European Journal of Operational Research*, 65, 4-19.

[15] DAGENAIS, C. (1992). *Automated staff scheduling: description of current system, definition of requirements, cost estimates.* Rapport interne, Hôpital Royal Victoria, Montréal, Québec, Canada.

[16] DANTZIG, G.B., BLATTNER W.O. et RAO, M.R. (1967). Finding a cycle in a graph with minimum cost to time ratio with application to a ship routing problem. *Theory of Graphs*, P. Rosenstiehl (Éditeur). Dunod (Paris) et Gordon et Breach (New-York), 77-84.

[17] DANTZIG, G.B. et WOLFE, P. (1960). Decomposition principle for linear programs. *Operations Research*, 8, 101-111.

[18] DESAULNIERS, G., DESROSIERS, J., IOACHIM, I., SOLOMON, M.M., SOUMIS F. et VILLENEUVE, D. (1997). A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. *Les Cahiers du GERAD*, G-94-46, École des Hautes Etudes Commerciales, Montréal, Québec, Canada, H3T 1V6.

[19] DESROCHERS, M. (1988). An algorithm for the shortest path problem with resource constraints. *Les Cahiers du GERAD*, G-88-27, École des Hautes Etudes Commerciales, Montréal, Québec, Canada, H3T 1V6.

[20] DESROCHERS, M. et SOUMIS, F. (1988). A generalized permanent labelling algorithm for the shortest path problem with time windows. *INFOR*, 26, 191-212.

[21] DESROCHERS, M. et SOUMIS, F. (1988). A reoptimization algorithm for the shortest path problem with time windows. *European Journal of Operational Research*, 35, 242-254.

[22] DESROSIERS, J., DUMAS, Y., SOLOMON M.M. et SOUMIS, F. (1994). Time constrained routing and scheduling. *Handbooks in Operations Research and Management Science*, Volume 8 on Network Routing, M.O. Ball*et al.* (Éditeurs.), Elsevier Science B.V., 35-139.

[23] DESROSIERS, J., PELLETIER P. et SOUMIS, F. (1983). Plus court chemin avec contraintes d'horaires. *RAIRO*, 17, 357-377.

[24] DESROSIERS, J., SOUMIS F. et DESROCHERS, M. (1984). Routing with time windows by column generation. *Networks*, 14, 545-565.

[25] DIJKSTRA, E.W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1, 269-271.

[26] DU MERLE O., VILLENEUVE, D., DESROSIERS, J. et HANSEN, P. (1997). Stabilisation dans le cadre de la génération de colonnes. *Les Cahiers du GERAD*, G-97-08, École des Hautes Etudes Commerciales, Montréal, Québec, Canada, H3T 1V6.

[27] DREYFUS, S.E. (1969). An appraisal of some shortest-path algorithms. *Operations Research*, 17, 395-412.

[28] FRISCH, I.T. (1963). Optimum routes in communication systems with channel capacities and channel reliabilities. *IEEE Trans. Commun. Systems*, CS-11, 241-244.

[29] GALLO, G. et PALLOTTINO, S. (1986). Shortest path methods: a unifying approach. *Math. Programming Studies*, 26, 38-64.

[30] GALLO, G. et PALLOTTINO, S. (1988). Shortest path algorithms. *Annals of Operations Research*, 13, 3-79.

[31] GALLO, G., PALLOTTINO, S., RUGGERI, C., STORCHI, G., (1982), Shortest paths: a bibliography. Quaderni SOFMAT, doc.num.81-P1-4-SOFMAT-27, CNR Roma, Italy.

[32] GAMACHE, M., SOUMIS, F., VILLENEUVE, D., DESROSIERS J. et E. GELINAS, (1997). The preférential bidding system at Air Canada. *Les Cahiers du*

GERAD, G-97-12, École des Hautes Etudes Commerciales, Montréal, Québec, Canada, H3T 1V6. A paraître dans *Transportation Science*.

[33] GILMORE, P.C. et GOMORY, R.E. (1961). A linear programming approach to the cutting stock problem. *Operations Research*, 9, 849-859.

[34] GILMORE, P.C. et GOMORY, R.E., (1963). A linear programming approach to the cutting stock problem: part II. *Operations Research*, 11, 863-888.

[35] HANDLER, G.Y. et ZANG, I. (1980). A dual algorithm for the constrained shortest path problem. *Networks*, 10, 293-310.

[36] HANSEN, P. (1980). Bicriterion path problems.*Multiple Criteria Decision Making: Theory and Applications*, Lecture Notes in Economics and Mathematical Systems, G. Fandel et T. Gal (Éditeurs), 177, 109-127.

[37] HANSEN, P., JAUMARD, B. et de ARAGÃO, M.P. (1991). Un algorithme primal de programmation linéaire généralisée pour les programmes mixtes. *Comptes-rendus de l'Académie des Sciences de Paris*, 313 (1), 557-560.

[38] HENIG, M. (1985). The shortest path problem with two objective functions. *European Journal of Operational Research*, 25, 281-291.

[39] HUNG, R. (1995). Hospital nurse scheduling.*Journal of Nursing Administration*, 25 (7/8) 21-23.

[40] IOACHIM, I., GELINAS, S., DESROSIERS J. et SOUMIS, F. (1997). A dynamic Programming algorithm for the shortest path problem with time windows and linear node costs. A paraître dans *Networks*, 1997.

[41] JAFFE, J. M. (1984). Algorithms for Finding paths with multiple constraints, *Networks*, 14, 95-116.

[42] JEYARATNAM, S. (1983). A new algorithm for finding the shortest path between a specified pair of nodes in a graph of nonnegative arcs. *European Journal of Operational Research*, 12, 375-378.

[43] JOKSCH, H.C. (1966). The shortest route problem with constraints. *Journal of Mathematical Analysis and Applications*, 14, 191-197.

[44] LOUI, R.P. (1983). Optimal paths in graphs with stochastic or multidimensional weights. *Communications of the Association of Computing Machinery*, 26 (9), 670-676.

[45] LEVINE, A. (1971.) Scheduling and fleet routing models for transportation systems. *Transportation Science*, 5, 232-255.

[46] MARTINS, E.Q.V. (1984). An algorithm to determine a path with minimal cost/capacity ratio. *Discrete Applied Mathematics*, 8, 189-194.

[47] MARTINS, E.Q.V. (1984). On a multicriteria shortest path problem. *European Journal of Operational Research*, 16, 236-245.

[48] MARTINS, E.Q.V. (1984). On a special class of bicriterion path problems. *European Journal of Operational Research*, 17, 85-94.

[49] MILLER, H.E., W.P. PIERSKALLA et G.J. RATH, (1976). Nurse scheduling using mathematical programming. *Operations Research*, 24(5), 857-871.

[50] MINOUX, M. (1975). Plus court chemin avec contraintes : algorithmes et applications. *Annal. Télecom.*, 30(11-12), 383-394.

[51] MOTE, J., MURTHY, I. et OLSON, D.L. (1991). A parametric approach to solving bicriterion shortest path problems. *European Journal of Operational Research*, 53, 81-92.

[52] OKADA, M. et OKADA, M. (1988). Prolog-based system for nursing staff scheduling implemented on a personal computer. *Computers and Biomedical Research*, 21, 53-63.

[53] OZKARAHAN, I. et BAILEY, J. (1988).Goal programming model subsystem of a flexible nurse scheduling support system. *IIE Transactions*, 306-316.

[54] PUNNEN, A.P. (1991). A linear time algorithm for the maximum capacity path problem. *European Journal of Operational Research*, 53, 402-404.

[55] RIBEIRO, C.C. et MINOUX, M. (1985). A heuristic approach to hard constrained shortest path problems. *Discrete Applied Mathematics*, 10, 125-137.

[56] TUNG, P.C.T. (1988). Finding a path with minimum cost to time ratio in a network. *Asia-Pacific Journal of Operational Research*, 3 43-52.

[57] TUNG, C.T. et CHEW, K.L. (1988). A bicriterion pareto-optimal path algorithm. *Asia-Pacific Journal of Operational Research*, 5, 166-172.

[58] TUNG, C.T. et CHEW, K.L. (1992). A multicriteria pareto-optimal path algorithm. *European Journal of Operational Research*, 62, 203-209.

[59] SAIGAL, R. (1968). A constrained shortest route problem, *Operations Research*, 16, 205-209.

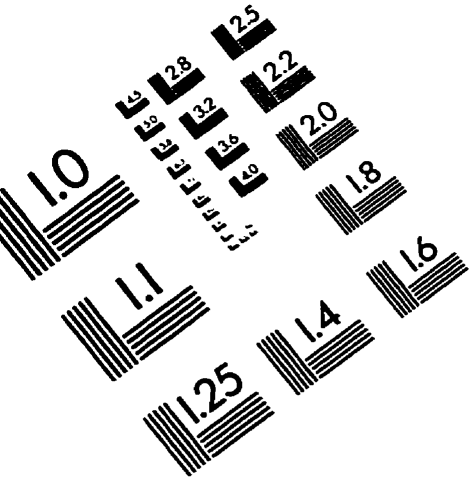
[60] SMITH, L.D. et WIGGINS, A. (1977). A computer based nurse scheduling system. *Computers and Operations Research*, 4, 195-212.

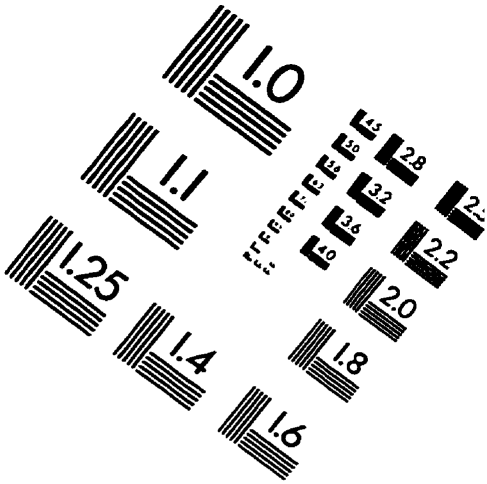
[61] VANDERBECK, F. et WOLSEY, L. A. (1994). An exact algorithm for IP column generation. *CORE Discussion Paper*, 9419, Université catholique de Louvain, Louvain-la-Neuve, Belgique.

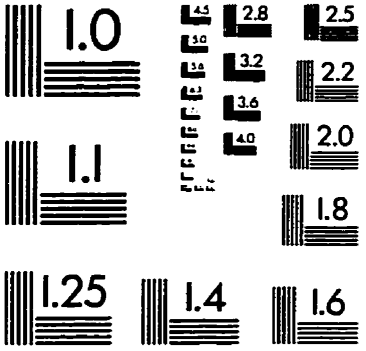
[62] WARNER, D.M. (1976).Scheduling nursing personnel according to nursing preference: a mathematical programming approach. *Operations Research*, 24(5), 842-856.

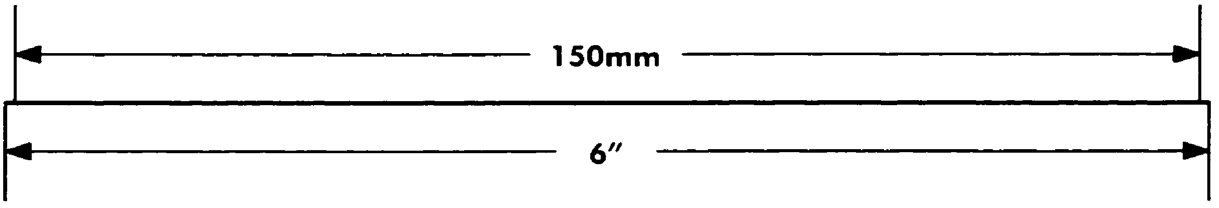
[63] WHITE, D.J. (1982). The set of efficient solutions for multiple objective shortest path problems, *Comput. Operations Research*, 9 (2), 101-107.

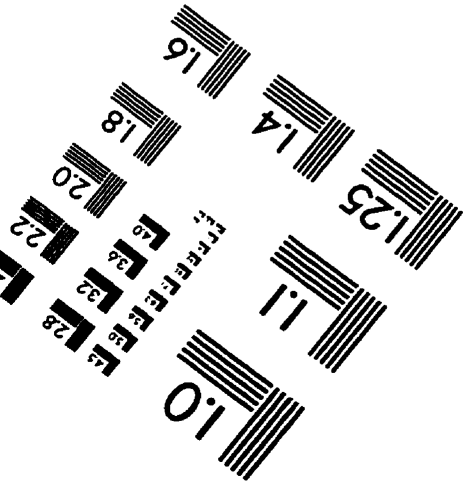
IMAGE EVALUATION TEST TARGET (QA-3)

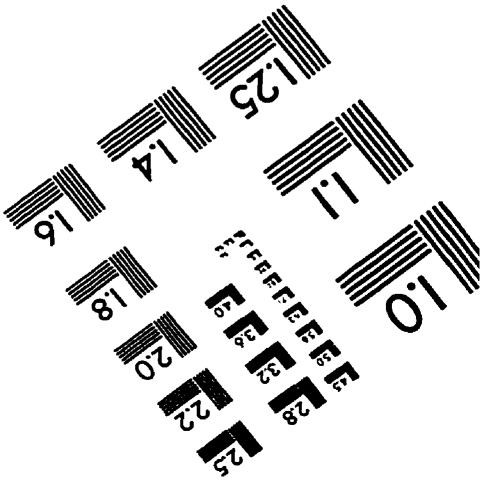












APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989